

1
2
3
4
5
6

Certified Tester Game Testing (CT-GaMe) Syllabus

Version 2.0

International Software Testing Qualifications Board



7
8
9
10

Provided by
Russian Software Testing Qualifications Board (RSTQB)



RSTQB

Russian Software Testing
Qualifications Board

11 Copyright Notice

12 Copyright Notice © International Software Testing Qualifications Board (hereinafter called ISTQB®)

13 ISTQB® is a registered trademark of the International Software Testing Qualifications Board.

14 Copyright © 2025, the authors Andrey Konushin (chair), Alexander Alexandrov, Alexander Torgovkin,
15 Vadim Lukovaty, Kseniya Kondakova, Jason Miller.

16 Copyright © 2022, the authors Andrey Konushin (chair), Alexander Alexandrov, Evgeny Glushkin,
17 Elizaveta Kruchinina, Vadim Lukovaty, Dmitry Melishev, Maxim Nikolaev, Alexander Prokhorov, Anton
18 Savvateev, Ayrat Sayfullov, Pavel Sharikov, Kirill Shevelev, Artyom Stukalov, Nikita Sysuev, Tatiana
19 Tepaeva, Alexander Torgovkin, Margarita Trofimova, Yaroslav Vereshchagin, Svetlana Yushina,
20 Lyubov Zhuravleva.

21 All rights reserved. The authors hereby transfer the copyright to the ISTQB®. The authors (as current
22 copyright holders) and ISTQB® (as the future copyright holder) have agreed to the following
23 conditions of use:

24 Extracts, for non-commercial use, from this document may be copied if the source is
25 acknowledged. Any Accredited Training Provider may use this syllabus as the basis for a
26 training course if the authors and the ISTQB® are acknowledged as the source and copyright
27 owners of the syllabus and provided that any advertisement of such a training course may
28 mention the syllabus only after official Accreditation of the training materials has been
29 received from an ISTQB®-recognized Member Board.

30 Any individual or group of individuals may use this syllabus as the basis for articles and
31 books, if the authors and the ISTQB® are acknowledged as the source and copyright owners
32 of the syllabus.

33 Any other use of this syllabus is prohibited without first obtaining the approval in writing of the
34 ISTQB®.

35 Any ISTQB®-recognized Member Board may translate this syllabus provided they reproduce
36 the Copyright Notice mentioned above in the translated version of the syllabus.

37

Revision History

38

Version	Date	Remarks
v1.0.1	October 21, 2022	GA Released
v.2.0 Alpha	October 18, 2025	Alpha version
v.2.0 Beta	February 25, 2026	Beta version Template v.4.4 used

39	Table of Contents		
40			
41	Copyright Notice.....		2
42	Revision History		3
43	Table of Contents.....		4
44	Acknowledgements		7
45	0.1 Purpose of this Syllabus		8
46	0.2 The Game Testing in Software Testing		8
47	0.3 Career Path for Testers		8
48	0.4 Business Outcomes		8
49	0.5 Learning Objectives and Cognitive Level of Knowledge		9
50	0.6 The Game Testing Certificate Exam		9
51	0.7 Accreditation		10
52	0.8 Handling of Standards		10
53	0.9 Level of Detail		10
54	0.10 How this Syllabus is Organized		10
55	1 The Specifics of Video Game Testing – 55 minutes		12
56	1.1 Video Game Testing Basics		12
57	1.1.1 The Specifics of Video Game Testing.....		12
58	1.1.2 Video Game Product Risks		13
59	1.1.3 Mitigating Video Game Risks by Testing		14
60	1.1.4 Game-Specific Failures		14
61	1.1.5 Specific Objectives of Video Game Testing		15
62	1.2 Typical Roles in a Video Game Development Team.....		16
63	1.3 Test Activities Throughout the Game Software Development Lifecycle		17
64	2 Testing Game Mechanics – 120 minutes		19
65	2.1 Principles and Concepts of Video Game Mechanics		19
66	2.1.1 Differences Between Testing Gameplay and Non-Gameplay Mechanics		20
67	2.1.2 Differences Between Testing Core and Meta Mechanics		21
68	2.1.3 Differences Between Testing Client-Side, Server-Side, and Client-Server Mechanics ..		22
69	2.1.4 Types of Game Mechanics Failures and Their Possible Causes		22
70	2.2 Video Game Mechanics Testing.....		23
71	2.2.1 Applying Fundamental Approaches to Video Game Mechanics Testing		23
72	3 Video Game Level Testing - 120 minutes		26
73	3.1 Principles and Concepts of Level Design in Video Games		26
74	3.1.1 The Term "Level" and Its Specifics		26
75	3.1.2 Types of Video Game Level Failures		27

76	3.2	Test Types in Video Game Levels Testing	30
77	3.2.1	Geometry Testing	30
78	3.2.2	Playtesting	31
79	3.3	Executing Video Game Level Testing	31
80	3.3.1	Applying Fundamental Approaches to Level Testing	31
81	4	Graphics Testing - 95 minutes	34
82	4.1	Principles and Concepts of Video Game Graphics	34
83	4.1.1	Features of Video Game Graphics	34
84	4.1.2	Types of Failures in Video Game Graphics	35
85	4.2	Test Types in Video Game Graphics Testing	41
86	4.2.1	Video Game Graphic Artistic Testing	41
87	4.2.2	Video Game Graphics Technical Testing	42
88	4.2.3	Video Game Graphics Playtesting	42
89	4.3	Executing Video Game Graphics Testing	42
90	5	Audio Testing - 130 minutes	45
91	5.1	Principles and Concepts of Audio Content in Video Games	45
92	5.1.1	Features of Video Game Audio	45
93	5.1.2	Video Game Audio Types	46
94	5.1.3	Video Game Audio Techniques	47
95	5.1.4	Types of Audio Failures	47
96	5.2	Test Types in Video Game Audio Testing	48
97	5.2.1	Video Game Content-Audio Testing	48
98	5.2.2	Video Game Audio Technical Testing	49
99	5.3	Executing Video Game Audio Testing	49
100	6	AI and Physics Testing - 170 minutes	51
101	6.1	Principles and Concepts of Artificial Intelligence (AI) and Physics in Video Games	51
102		Video Game Artificial Intelligence Features	51
103	6.1.1	51	
104	6.1.2	Types of AI Failures in Video Games	52
105	6.1.3	Video Game Physics Features	53
106	6.1.4	Types of Game Physics Failures	54
107	6.2	Executing AI and Game Physics Testing	55
108	6.2.1	Executing AI Testing in Video Games	55
109	6.2.2	Executing Game Physics Testing	56
110	7	Localization Testing - 150 minutes	58
111	7.1	Principles and Concepts of Video Game Localization	58
112	7.1.1	Video Game Localization Essentials	58

113	7.1.2	Features of Video Game Localization	59
114	7.1.3	Full and Partial Localization	60
115	7.1.4	Root Causes of Localization Defects	61
116	7.1.5	Video Game Localization Failures	61
117	7.2	Test Types in Localization Testing	63
118	7.2.1	Linguistic Testing and Adaptation Testing	64
119	7.2.2	Localization Technical Testing	64
120	7.3	Executing Localization Testing	65
121	8	Game Controllers Testing - 55 minutes	67
122	8.1	Principles and Concepts of Video Game Controllers	67
123	8.1.1	Types of Video Game Controllers	67
124	8.1.2	Failures Related to the Performance of Video Game Controllers	68
125	8.2	Test Types in Game Controller Testing	69
126	8.2.1	Interoperability Testing	70
127	8.2.2	Performance Testing	70
128	8.2.3	Ergonomics Testing	70
129	8.2.4	Testing Video Game Controllers for Compliance	71
130	8.2.5	Usability Testing	71
131	9	References	72
132	9.1	Standards	72
133	9.2	ISTQB documents	72
134	9.3	Books	72
135	9.4	Links (Web/Internet)	73
136	10	Appendix A – Learning Objectives/Cognitive Level of Knowledge	74
137	11	Appendix B – Business Outcomes Traceability Matrix with Learning Objectives	76
138	12	Appendix C – Release Notes	79
139	13	Appendix D – Game Testing Specific and other Terms	80
140	14	Appendix E – Index	83
141			

142 Acknowledgements

143 This document was formally released by the General Assembly of the ISTQB® on <date>

144 It was produced by a core team of the Russian Software Testing Qualifications Board (RSTQB):
145 Andrey Konushin (President of the Board), Alexander Alexandrov (Editor-in-Chief), Alexander
146 Torgovkin, Vadim Lukovaty, Jason Miller.

147 The core team thanks the review team for their suggestions and input. Russian Software Testing
148 Qualifications Board would like to acknowledge and thank LLC "Bytex" for their contribution to the
149 development of this syllabus.

150 In addition, the core team would like to acknowledge and thank the leaders of the Working Groups for
151 their early and ongoing guidance: Galit Zucker (General Secretary), Florian Fieber (chair, Testing in
152 Particular Domains Working Group), Richard Seidl (vice-chair, Testing in Particular Domains Working
153 Group), Matthias Hamburg (chair, Glossary Working Group) and Klaus Skafté (chair, Exam Working
154 Group).

155 The following persons participated in the reviewing, commenting and balloting of this syllabus:

Kseniya Kondakova
Iliia Kulakov
Murian Song
Reifa Tangon
Shinsuke Matsuki
Andrea Horváth
Mitko Mitev

Emiola Ibironke
Matthias Hamburg
Klaus Skafté
Vipul Kocher
Albert Laura
Lukas Piska
Aneta Derková

Pavel Sharikov
Cristina Sobrero
Kenji Onishi
Satomi Yoshizawa
Gary Mogyorodi
Péter Földházi
Michael Stahl

156

157 0.1 Purpose of this Syllabus

158 This syllabus forms the basis for the ISTQB® Certified Tester Game Testing. The ISTQB® provides
159 this syllabus as follows:

160 1. To National Boards, to translate into their local language and to accredit training providers.
161 National Boards may adapt the syllabus to their particular language needs and modify the
162 references to adapt to their local publications.

163 2. To exam providers, to derive examination questions in their local language adapted to the
164 learning objectives for each syllabus.

165 3. To training providers, to produce courseware and determine appropriate teaching methods.

166 4. To certification candidates, to prepare for the exam (as part of a training course or
167 independently).

168 5. To the international software and systems engineering community, to advance the profession
169 of software and systems testing, and as a basis for books and articles.

170 The ISTQB® may allow other entities to use this syllabus for other purposes, provided they seek and
171 obtain prior written permission.

172 0.2 The Game Testing in Software Testing

173 The Certified Tester Game Testing qualification is aimed at anyone involved in software testing who
174 wishes to broaden their knowledge of game testing or anyone who wishes to start a specialist career in
175 game testing. The qualification is also aimed at anyone involved in game software engineering who
176 wishes to gain a better understanding of game testing.

177 The syllabus considers the following principal aspects of game testing:

- 178 • Technical aspects,
- 179 • Method-based aspects,
- 180 • Organizational aspects.

181 0.3 Career Path for Testers

182 Building on the Foundation Level, the Game Testing supports the definition of career paths for
183 professional testers. A person with the Game Testing certificate will have extended the broad
184 understanding of testing acquired at the Foundation Level to enable him or her to work effectively as a
185 professional tester in a game project.

186 Those possessing a Game Testing certificate may use the acronym CT-GaMe.

187 Please visit [ISTQB-Web] for the latest overview of ISTQB®'s career path.

188 0.4 Business Outcomes

189 This section lists the Business Outcomes expected of a candidate who has achieved the Game Testing
190 certification.

191 A Game Testing Certified Tester can...

GaMe-BO1	Describe basic concepts of video game testing
----------	---

GaMe-BO2	Determine video game risks, goals and requirements under the needs and expectations of stakeholders
GaMe-BO3	Conceptually design, implement and execute basic video game tests
GaMe-BO4	Know the approaches to video game testing and their purpose
GaMe-BO5	Understand the approaches to video game testing and their purpose
GaMe-BO6	Determine how test activities align with the software development lifecycle and reduce the cost of developing and publishing video games

192

193 0.5 Learning Objectives and Cognitive Level of Knowledge

194 Learning Objectives support the business outcomes and are used to create the Game Testing exams.

195 In general, all contents of this syllabus are examinable, except for the Introduction and Appendices.
196 The exam questions will confirm knowledge of keywords at K1 level (see below) or learning objectives
197 at the respective level of knowledge.

198 The specific learning objectives and their levels of knowledge are shown at the beginning of each
199 chapter, and classified as follows:

- 200 • K1: Remember
- 201 • K2: Understand
- 202 • K3: Apply

203 Further details and examples of learning objectives are given in Appendix A.

204 For all terms listed as keywords just below chapter headings, the correct name and definition from the
205 ISTQB® glossary shall be remembered (K1), even if not explicitly mentioned in the learning objective.

206 0.6 The Game Testing Certificate Exam

207 The Game Testing Certificate exam will be based on this syllabus. Answers to exam questions may
208 require the use of material based on more than one section of this syllabus. Standards and books are
209 included as references, but their content is not examinable, beyond what is summarized in the syllabus
210 itself from such standards and books.

211 Refer to [ISTQB_EXAM_S&R] Exam Structures and Rules for the Game Testing V2.0 document for
212 further details.

213 The entry criterion for taking the ISTQB® Certified Tester Game Testing exam is that candidates are
214 interested in software testing. However, it is strongly recommended that candidates also:

- 215 • have at least a minimal background in either game software development or testing, such as 1
216 year experience as a game software user acceptance tester.
- 217 • take a course that has been accredited to ISTQB® standards (by one of the ISTQB-recognized
218 member boards).

219 Entry Requirement Note: The ISTQB® Foundation Level certificate shall be obtained before taking the
220 ISTQB® Certified Tester Game Testing certification exam.

221 0.7 Accreditation

222 An ISTQB® Member Board may accredit training providers whose course material follows this syllabus.
223 Training providers should obtain accreditation guidelines from the Member Board or body that performs
224 the accreditation. An accredited course is recognized as conforming to this syllabus and is allowed to
225 have an ISTQB® exam as part of the course.

226 The accreditation guidelines for this syllabus follow the general Accreditation Guidelines published by
227 the Processes Management and Compliance Working Group.

228 0.8 Handling of Standards

229 International standardization organizations like IEEE and ISO have issued standards associated with
230 quality characteristics and software testing. Such standards are referenced in this syllabus. The purpose
231 of these references is to provide a framework (as in the references to ISO 25010 regarding quality
232 characteristics) or to provide a source of additional information if desired by the reader. Please note
233 that ISTQB® syllabi are using the standard documents as reference. Standards documents are not
234 intended for examination. Refer to section 9. References for more information on Standards.

235 0.9 Level of Detail

236 The level of detail in this syllabus allows internationally consistent courses and exams. To achieve this
237 goal, the syllabus consists of:

- 238 • General instructional objectives describing the intention of the Certified Tester Game Testing
239 Level
- 240 • A list of terms that students must be able to recall
- 241 • Learning objectives for each knowledge area, describing the cognitive learning outcome to be
242 achieved
- 243 • A description of the key concepts, including references to sources such as accepted literature
244 or standards

245 The syllabus content is not a description of the entire knowledge area of game testing; it reflects the
246 level of detail to be covered in Certified Tester Game Testing training courses. It focuses on test areas
247 and techniques that can apply to most game projects.

248 The syllabus uses the terminology (i.e. the name and meaning) of the terms used in software testing
249 and quality assurance according to the ISTQB® Glossary.

250 0.10 How this Syllabus is Organized

251 The Certified Tester Specialist Game Testing syllabus contains eight chapters covering the knowledge
252 necessary to be a Game Testing Specialist.

253 The top-level heading for each chapter specifies the minimum time for the chapter; timing is not provided
254 below chapter level. For accredited training courses, the syllabus requires a minimum of 14 and 5/6
255 hours of instruction, distributed across the eight chapters as follows:

- 256 • Chapter 1: 55 minutes The Specifics of Video Game Testing
 - 257 ○ The tester learns the basic principles related to video game testing, the reasons why
258 video game testing is required, what the test objectives are, and the difference between
259 video game testing and playing the video game.
 - 260 ○ The tester learns about video game sub-systems.
 - 261 ○ The tester understands the game test process, the major activities, and work products.

- 262 • Chapter 2: 120 minutes Testing Game Mechanics
- 263 ○ The tester learns to distinguish between different game mechanics and its test
- 264 approaches.
- 265 ○ The tester learns both dynamic and static test techniques used for game mechanics
- 266 testing.
- 267 • Chapter 3: 120 minutes Video Game Level Testing
- 268 ○ The tester learns the dependency of the level design and test types and its tool support.
- 269 ○ The tester learns about typical failures and their causes.
- 270 ○ The tester learns about test types in video game level testing.
- 271 • Chapter 4: 95 minutes Graphics Testing
- 272 ○ The tester learns about features and types of the graphic content of the video game.
- 273 ○ The tester learns about typical failures and their causes.
- 274 ○ The tester learns about test types in video game graphics testing.
- 275 • Chapter 5: 130 minutes Audio Testing
- 276 ○ The tester learns about types of the sound content of the video game.
- 277 ○ The tester learns about typical failures and their causes.
- 278 ○ The tester learns about test types in video game sound testing.
- 279 • Chapter 6: 170 minutes AI and Physics Testing
- 280 ○ The tester learns about types of AI and physics of the video game.
- 281 ○ The tester learns about typical failures and their causes.
- 282 ○ The tester learns about test types in video game AI and physics testing.
- 283 • Chapter 7: 150 minutes Localization Testing
- 284 ○ The tester learns the basics and differences of localization and internationalization, its
- 285 risks.
- 286 ○ The tester learns about typical failures and their causes.
- 287 ○ The tester learns about test types in video game localization testing.
- 288 • Chapter 8: 55 minutes Game Controllers Testing
- 289 ○ The tester learns about types of game controllers including related failures and their
- 290 possible causes.

291

292 It is important to note that the terms game, computer game, and video game are used synonymously
293 within this syllabus. This verifies readability and consistency throughout the training program. A game
294 is considered strictly as software and should not be confused with sports or gambling activities.

295 It is advisable to those readers who have a little or no prior experience in video game testing to refer to
296 the Appendix D before reading further.

297

298 **1 The Specifics of Video Game Testing – 55 minutes**

299 **Keywords**

300 Compatibility, functional suitability, localization, risk

301 **Video Game Specific Keywords**

302 Concept phase, game designer, gameplay, gameplay mechanics, immersion, level design,
303 mechanics, multi-platform, narrative, non-gameplay, pre-production phase, post-production phase,
304 production phase, quest, racing wheel, video game

305

306 **Learning Objectives for Chapter 1**

307 **1.1 Video Game Testing Basics**

308 GaMe-1.1.1 (K1) Recognize specifics of video game testing

309 GaMe-1.1.2 (K1) Recall product risks in video game software

310 GaMe-1.1.3 (K1) Recall how video game risks can be mitigated by testing.

311 GaMe-1.1.4 (K2) Give examples of specific failures related to video game testing

312 GaMe-1.1.5 (K2) Explain the specific test objectives of video game testing

313 **1.2 Typical Roles in a Video Game Development Team**

314 GaMe-1.2.1 (K1) Recognize specific roles and tasks in the video game development team

315 **1.3 Test Activities throughout the Game Software Development Lifecycle**

316 GaMe-1.3.1 (K1) Recall test activities throughout the video game software development lifecycle

317 **1.1 Video Game Testing Basics**

318 **1.1.1 The Specifics of Video Game Testing**

319 As a software product, a video game has several distinguishing characteristics compared to other types
320 of software. Unlike many standalone applications, a video game is often implemented as multiple
321 interacting software applications and services that operate together to deliver the complete game
322 experience. From a testing perspective, this means that game behavior may depend on interactions
323 between client-side applications, server-side components, and supporting services, rather than a single
324 executable system.

325 These characteristics stem from the presence of specific subsystems, which include:

- 326 • Graphics subsystem
- 327 • Audio subsystem
- 328 • Game logic and mechanics subsystem
- 329 • Game physics subsystem
- 330 • Artificial intelligence (AI) subsystem

331 Additionally, video games possess unique attributes that define them as a distinct class of software,
332 such as:

- 333 • Ensuring fairness, preventing any group or individual from gaining an unfair advantage through
334 software exploitation.

- 335
- Compatibility with various input devices (controllers) to enable intuitive gameplay interaction.
- 336
- Multi-platform and cross-platform compatibility.
- 337
- Adherence to real-world or historical prototypes (characters, objects, events).
- 338
- Complex localization processes that go beyond simple language translation.

339 Due to the interconnected nature of these subsystems and features, defects that emerge in video
340 games are often specific to this software category.

341 Another fundamental aspect of game testing is the need to validate not only functional suitability but
342 also the gameplay. Understanding deviations in expected game behavior requires testers to have deep
343 domain knowledge across multiple disciplines. A game tester must assess balance, difficulty, narrative
344 consistency, and the interaction of game components. Thus, this role demands not only technical skills
345 but also creative and analytical thinking.

346 1.1.2 Video Game Product Risks

347 The rapid growth of the gaming industry has led to the expansion of games across multiple platforms.
348 At the same time, player expectations have risen, making quality a critical success factor.

349 Since video games are developed within structured software projects, they share common product risks
350 with other types of software. However, due to their unique characteristics, video games are subject to
351 additional specific risks, such as:

352 **Product complexity**

- 353
- Cheating and unfair advantages
- 354
- Market success dependency on subjective player opinions
- 355
- Challenges in cross-platform development and controller compatibility
- 356
- Uncertainty in predicting the effectiveness of game mechanics

357 **Game Complexity**

358 All software systems are inherently complex, but video games introduce a unique challenge. Each game
359 subsystem relies on specialized hardware capable of supporting advanced rendering, physics
360 simulations, and real-time computations. These systems must work together seamlessly to deliver an
361 engaging player experience.

362 **Cheating and Exploits**

363 Cheating represents a major product risk that can damage a game's reputation and commercial
364 success. Unfair advantages gained by exploiting vulnerabilities can lead to player frustration, decreased
365 retention rates, and negative community feedback. If cheating is widespread, it can erode trust in the
366 game and deter new players from joining.

367 **Subjectivity of Player Perception**

368 Unlike business applications, where functional suitability is the primary factor for success, game quality
369 is largely determined by user engagement and entertainment value. Players may tolerate minor
370 functional defects but they will abandon a game if it feels boring, repetitive, or visually outdated.

371 Many purchase decisions are influenced by game reviews, critics, and community sentiment, making
372 negative public perception a significant risk.

373

374

375 **Cross-Platform Challenges**

376 Multi-platform compatibility is a common practice, but ensuring consistent behavior and user experience
377 across different hardware, operating systems, and input methods presents a significant product risk.
378 Differences in platform capabilities, performance characteristics, and control schemes can lead to
379 platform-specific defects or inconsistent gameplay behavior.

380 Modern gaming platforms also support specialized controllers (e.g., racing wheels, VR motion
381 controllers), each requiring unique handling in input processing and user experience design. As a result,
382 testers must consider the increased risk of platform and device specific failures, and apply appropriate
383 test selection and prioritization to address the most critical usage scenarios.

384 **Game Mechanics Effectiveness**

385 Game mechanics define rules and interactions within a game. Predicting how players will perceive and
386 engage with specific mechanics is difficult, making this a high-risk area.

387 1.1.3 Mitigating Video Game Risks by Testing

388 Most modern video games, particularly open-world and live-service games, are too complex to be fully
389 tested under all possible object interactions, events, and variables. As a result, different categories of
390 risk must be mitigated using different testing activities.

391 Analytical and tool-supported testing activities help mitigate risks related to large volumes of data,
392 frequent updates, and regression across interconnected systems. Experience-based testing activities,
393 such as playtesting and usability testing, help mitigate risks related to subjective player perception,
394 balance, and overall gameplay experience. Non-functional testing activities address risks related to
395 performance, stability, and compatibility across platforms and environments.

396 Risk-based testing supports this approach by helping testers prioritize which risks require deeper or
397 more frequent testing based on their likelihood and potential impact.

398 For each condition, two key questions determine priority:

- 399 1. How likely is a player to encounter this defect?
- 400 2. If encountered, what is the impact on the player's experience and business?

401 For example, in linear progression games, defects occurring early in the game may be prioritized over
402 those appearing after 50+ hours of playtime, as early- phase defects can drive away new players.

403 Testers focus on defect clusters rather than isolated defects, allowing developers to resolve systemic
404 defects at the code or architectural level to enhance overall quality.

405 Player onboarding and tutorial phases are especially critical, as first impressions determine long-term
406 retention.

407 Another crucial factor in mitigating risk is early involvement of test team in the development process.
408 When testers participate in the design and prototyping phases, they can identify potential risk areas
409 before code is even written. This proactive approach helps shape features with testability in mind,
410 reduces costly rework, and verifies that quality considerations are embedded into the game's foundation
411 — not retrofitted at the end.

412 1.1.4 Game-Specific Failures

413 Most enterprise or business software is designed with utilitarian objectives, such as processing data,
414 managing transactions, or supporting organizational workflows. When such systems — for example,

415 accounting applications or online tax platforms — encounter defects or experience outages, users may
416 face inconvenience and organizations may suffer financial or regulatory consequences. However, the
417 overall stakeholder response is typically procedural and rooted in risk management.

418 In contrast, video game software serves an entertainment function and often carries a strong emotional
419 component. Games represent immersive experiences that deliver escapism, interactivity, and narrative
420 engagement. As a result, failures in game software — particularly those released after extensive public
421 promotion — can provoke strong emotional reactions. Users may not only express dissatisfaction as
422 consumers but also react with personal disappointment, reflecting the perceived breach of trust and
423 expectations established by pre-release marketing and community engagement.

424 Given the unique complexity of video games, certain failure types are specific to this software category,
425 including:

- 426 • 3D model and animation failures, including physics-based interactions.
- 427 • Gameplay mechanics anomalies and unintended interactions.
- 428 • Level design defects, such as incorrect collisions or unreachable areas.
- 429 • Audio synchronization and sound failures.
- 430 • Game world physics inconsistencies (e.g., floating objects, unrealistic gravity).
- 431 • AI behavior errors, affecting non-player characters (NPCs).
- 432 • Narrative or quest logic failures, leading to broken story progression.
- 433 • Cultural, legal, and ethical localization failures.
- 434 • Controller and input device compatibility failures.

435 Since games are interactive experiences, failures directly impact immersion and player satisfaction.
436 Unlike productivity software, where usability and efficiency are paramount, game testing focuses on
437 maintaining engaging and fair gameplay.

438 1.1.5 Specific Objectives of Video Game Testing

439 Video game testing differs from any other software testing due to the unique characteristics of video
440 games as interactive entertainment products.

441 Video games consist of multiple integrated subsystems, each requiring specialized test approaches.

442 Testing in games often involves qualitative assessment. Components like difficulty balance, fun factor,
443 narrative coherence, and emotional impact must be considered. Unlike typical software, minor
444 functional defects may be tolerated, but poor gameplay leads to rejection by users.

445 Visual presentation is critical in games. Testers must evaluate animations, transitions, lighting, camera
446 behavior, and VFX (visual effects) fidelity. Failures related to clipping, frame rate drops, or incorrect
447 rendering can severely affect immersion.

448 Video games are typically released on multiple platforms (PC, consoles, mobile, VR (virtual reality)),
449 each with its own hardware constraints and control schemes (e.g., keyboard/mouse, gamepads, motion
450 sensors). Evaluating consistency and responsiveness across platforms is a core test objective.

451 Video games often support multiple languages and markets, requiring extensive localization testing.
452 This includes not only translated text but also voice-overs, legal compliance, cultural sensitivity, and
453 layout adjustments.

454 Video game testers must blend analytical test methods with creative thinking. They often need to
455 simulate player behavior, uncover unintended consequences of mechanics, and provide feedback on
456 design coherence.

457 A common misconception is that game testers spend their time simply playing games. In reality, testers
458 conduct structured testing based on defined test cases, specifications, and quality standards.

459 While casual players engage for entertainment, testers systematically verify compliance with
460 requirements, often replaying the same scenario multiple times to validate different aspects of the game.

461 Repetitive tasks and manual regression testing may seem monotonous, but identifying game-breaking
462 failures is highly rewarding and crucial for delivering a polished product.

463 **Connection with Other Syllabi**

464 The approaches discussed in this syllabus apply exclusively to game testing. They are relevant to both
465 software testing (e.g., testing game mechanics) and hardware testing (e.g., testing controllers).

466 Some aspects of game testing are not significantly different from testing other software, especially when
467 performing test types such as regression testing, performance testing, or usability testing. Core testing
468 concepts, non-functional test types, and the specifics of testing software on mobile platforms are
469 covered in detail in other ISTQB® syllabi.

470 **1.2 Typical Roles in a Video Game Development Team**

471 In most cases, a video game development team comprises various roles, many of which are similar to
472 those found in other areas of software development. In smaller teams, one individual may take on
473 multiple roles simultaneously. Below is a summary of the key responsibilities of typical roles in a game
474 development project. Additional roles specific to certain areas of game development are covered in
475 relevant sections of this syllabus.

Game Functionality	Role	Responsibilities
Story, game mechanics, game balance.	Game designer / Narrative designer	Directs the game's creative vision, designs storylines, game mechanics, balance, reward systems, monetization strategies.
Graphics subsystem: 2D assets	Artist	Creates characters and assets, produces animation sprites.
Graphics subsystem: 3D assets	3D Modeler / Rigger / Skinner	Develops 3D character models and assets, creates skeletal rigs for animation, textures models.
Graphics subsystem: animation	Animator	Animates in-game objects and characters.
Graphics subsystem: level design	Level designer	Designs game level architecture, ensures compliance with game rules at each level.
Graphics subsystem: user interface	UI (User Interface) designer	Develops graphical interfaces to enhance user interaction with the game and its subsystems.
Graphics subsystem: visual effects	VFX specialist	Creates various in-game visual effects.

Physics subsystem	Game engine programmer	Implements physics simulation within the game engine.
Artificial Intelligence (AI) subsystem	AI programmer	Writes scripts to simulate realistic AI behavior for in-game characters.
Data storage subsystem	Server-side developer	Develops functionality for storing player data on servers.
Software subsystem: interfaces	Interface programmer	Implements user interfaces to facilitate intuitive interaction with the game and its systems.
Audio subsystem	Audio engineer	Records and processes game-related sound effects.
Audio subsystem	Composer	Composes and records background music and soundtracks for the game.
Game product as a whole	Producer	Manages game development from a business perspective.
Game product as a whole	Localization specialist, Consultant, Translator, Voice actor	Ensures proper localization of the game product for specific markets.
Game product as a whole	Tester	Identifies defects across all game subsystems and evaluates the quality of the subsystems as well as the complete video game.
Game product as a whole	Business owner	Defines the business development strategy.

476

477 1.3 Test Activities Throughout the Game Software Development 478 Lifecycle

479 The video game development lifecycle typically consists of several key phases:

480 **Concept Phase**

481 The main goal of this phase is to establish a unified vision of the future product between the
482 development team and stakeholders. Testware created in this phase include:

- 483 • Test approach draft
- 484 • Test plan draft
- 485 • High-level test effort estimation
- 486 • Test environment outline

487

488 **Pre-Production Phase**

489 The primary objective of this phase is to develop a prototype—the initial working version of the game.
490 The technical feasibility of the project is also assessed.

491 Game testers and other development team members review the prototype to evaluate gameplay
492 mechanics and identify early-phase technical challenges. Testware created and updated in this phase
493 include:

- 494 • Test approach
- 495 • Test plan
- 496 • Test schedule
- 497 • Test environment setup

498 **Production Phase**

499 During this phase, the full game product is developed. This is the longest phase and is often divided
500 into milestones for alpha and beta releases.

501 Testware created and updated in this phase include:

- 502 • Test plan
- 503 • Test schedule
- 504 • Test conditions, checklists, test cases, automated test scripts, test procedures, test suites
- 505 • Defect reports
- 506 • Test completion reports

507 **Post-Production Phase**

508 The post-production phase includes maintenance test activities that support the software development
509 throughout its lifecycle. During this phase, updates and patches undergo testing and regression testing
510 is conducted, as described in [ISTQB_FL_SYL].

511

512 **2 Testing Game Mechanics – 120 minutes**

513 **Keywords**

514 Black-box testing, canary testing, defect, failure, functional completeness, functional correctness,
515 playtesting

516 **Video Game Specific Keywords**

517 concept phase, core mechanics, game environment, meta mechanics, player versus player (PvP),
518 pre-production phase, production phase, server-side mechanics, store
519

520 **Learning Objectives for Chapter 2**

521 **2.1 Principles and Concepts of Video Game Mechanics**

522 GaMe-2.1.1 (K2) Differentiate the testing of gameplay mechanics and non-gameplay mechanics

523 GaMe-2.1.2 (K2) Differentiate the testing of core mechanics and meta mechanics

524 GaMe-2.1.3 (K2) Differentiate the testing of client-side, server-side, and client-server mechanics

525 GaMe-2.1.4 (K2) Give examples of failures in video game mechanics

526 **2.2 Video Game Mechanics Testing**

527 GaMe-2.2.1 (K3) Apply test types and test approaches to game mechanics testing

528

529 **2.1 Principles and Concepts of Video Game Mechanics**

530 Mechanics are the foundation of any game. They serve as a means of interaction between the game
531 and the user. Mechanics take into account the influence and feedback between the game and the user,
532 as well as changes in the video game state within predefined requirements.

533 Game mechanics largely determine the player's experience, impression, and level of engagement.
534 Since game mechanics are at the core of any game, verifying their correct operation is typically a top
535 priority for testers, though priority should be determined through risk-based testing considering business
536 impact, player experience, and technical dependencies. Failures found during such testing are typically
537 among the most critical.

538 A defect in a core mechanics, causing the failure of a jump action, can make it impossible to overcome
539 obstacles and, consequently, complete the game.

540 Even if a mechanics is rarely used or is unrelated to the game's progression (e.g., a character's ability
541 to ride a horse), its incorrect operation can negatively impact the overall perception of the game.

542 Game mechanics must not only function correctly but also remain consistent and intuitive throughout
543 the game. Inconsistent behavior, such as variable jump height or unpredictable collision detection, can
544 frustrate players and break immersion. That's why testers often develop specific scenarios to validate
545 edge cases and evaluate if the mechanics behave reliably under different conditions.

546 Moreover, mechanics often interact with each other — for instance, combining a sprint function with a
547 climbing mechanic. Testing these interactions is crucial, as defects may not surface during isolated

548 tests but appear when features are combined. Automated test tools are used to verify functional
549 correctness and performance, while playtesting assesses how mechanics feel and function in real
550 gameplay from a player experience perspective. Ultimately, ensuring smooth and reliable mechanics
551 significantly enhances user satisfaction and gameplay fluidity.

552 **Types of Mechanics in Video Games**

553 Mechanics can be classified based on different aspects and objectives of the game:

- 554 • Based on the type of player interaction with the mechanics:
 - 555 ○ Gameplay Mechanics
 - 556 ○ Non-Gameplay Mechanics
- 557 • Based on their influence on core gameplay:
 - 558 ○ Core Mechanics
 - 559 ○ Meta Mechanics
- 560 • Based on game architecture:
 - 561 ○ Client-Side Mechanics
 - 562 ○ Server-Side Mechanics
 - 563 ○ Client-Server Mechanics

564 2.1.1 Differences Between Testing Gameplay and Non-Gameplay Mechanics

565 Testing gameplay and non-gameplay mechanics is a part of functional testing and is conducted using
566 test approaches, test techniques, and tools described in [ISTQB_FL_SYL]. However, there are key
567 differences in how they are tested.

568 **Gameplay Mechanics**

569 Gameplay mechanics are elements that directly influence player progress toward game objectives and
570 form the core interactive experience. These mechanics may be directly controlled by the player (e.g.,
571 character movement, shooting) or may affect gameplay indirectly (e.g., enemy AI, environmental
572 hazards). While many gameplay mechanics provide visible feedback, some operate with hidden
573 calculations (e.g., damage formulas, random generation).

574 Testing gameplay mechanics usually has a higher priority than testing non-gameplay mechanics
575 because they are more noticeable to players. Failures in these mechanics are likely to be observed and
576 can lead to negative feedback from the gaming community.

577 Gameplay mechanics require verification of functional correctness and functional completeness and
578 validation of functional appropriateness. In addition to functional testing, usability and UI testing are also
579 conducted.

580 A tester can often predict how a gameplay mechanics should function and its expected results, even if
581 documentation describing the mechanics is incomplete.

582 **Non-Gameplay Mechanics**

583 Non-gameplay mechanics are typically hidden from the user, although they may be triggered by player
584 actions. For example, an in-game purchase may initiate the following sequence of non-gameplay
585 mechanics:

- 586 1. A request is sent to the database to retrieve account data.
- 587 2. A transaction is performed to deduct funds.
- 588 3. The purchase is logged.
- 589 4. A confirmation email is sent to the user.
- 590 5. The purchase date and the item type are recorded for future promotional offers.

591 Testing non-gameplay mechanics primarily focuses on verification of functional correctness and
592 functional completeness. While failures in these mechanics may not directly affect the player, they can
593 distort game analytics, leading to incorrect business decisions and potential player churn.

594 Unlike gameplay mechanics, testing non-gameplay mechanics requires detailed requirements
595 specifications or predefined test cases, since they are often implemented at the developer's request
596 and vary across games.

597 2.1.2 Differences Between Testing Core and Meta Mechanics

598 **Core Mechanics**

599 Core mechanics define the fundamental game functionalities and are responsible for providing players
600 with the intended experience. They categorize a game into a specific genre.

601 Examples of core mechanics:

- 602 • Jumping (Platformer game)
- 603 • Building structures (Real Time Strategy game)
- 604 • Shooting (FPS game)

605 Removing a core mechanics can entirely change the game's genre. For example, removing vehicle
606 movement from a racing simulator would eliminate its classification as a racing game.

607 **Meta Mechanics**

608 Meta mechanics are gameplay elements that enhance the gaming experience but are not essential to
609 defining the game's genre. Removing a meta mechanic does not change what type of game it is, though
610 it may reduce player engagement or variety.

611 Examples of meta mechanics:

- 612 • Choosing a car model based on track type in a racing game - not a mandatory action, but it can
613 increase the chance of winning a race
- 614 • Achievement tracking systems - provide additional goals but don't change core gameplay
- 615 • Collecting collectibles items - increases player retention but does not affect the core gameplay

616 Energy replenishment after a certain amount of time in match-3 game - motivates the player to log in
617 more often. **Testing Core and Meta Mechanics**

618 Core mechanics often have stricter performance efficiency requirements because delays can directly
619 impact gameplay objectives and player success. For instance, if weapon switching in combat (a core
620 mechanics) is slow, a player may lose a battle.

621 In contrast, the time behavior of a meta mechanics is usually less critical. For example, a slight delay
622 in changing a character's appearance may be classified as a low-priority defect, since it doesn't affect
623 the ability to achieve game objectives.

624 The distinction between core and meta mechanics also influences how defects are prioritized during
625 testing. Since core mechanics directly affect gameplay flow and the ability to achieve game objectives,
626 bugs in these systems often require immediate attention and validation. On the other hand, defects in
627 meta mechanics, while still important for overall user satisfaction, may be deprioritized if they do not
628 disrupt critical gameplay functions. This prioritization ensures that development and test efforts align
629 with the player's most essential interactions, maintaining a smooth and engaging experience.

630 The differences between core and meta mechanics also impact the timing of their testing. For example,
631 core mechanics are typically tested earlier and more frequently throughout development due to their
632 critical nature. More details on testing at different development phases are discussed in section 2.3.1.

633 2.1.3 Differences Between Testing Client-Side, Server-Side, and Client-Server 634 Mechanics

635 The need to test client-side, server-side, and client-server mechanics depends on the game's
636 architecture. Each type of mechanic requires different test approaches based on its implementation.

637 **Client-Side Mechanics**

638 Client-side mechanics are processed on the player's device, meaning all relevant data is stored locally.
639 Some games function entirely on the client side, eliminating the need for a server. Such games are
640 typically single-player and do not require an internet connection.

641 Even in multiplayer games, certain mechanics remain client-side, such as viewing the game map or
642 opening the inventory. These are tested applying functional testing without server interaction.

643 Client-side mechanics are generally tested using black-box testing, often involving UI testing.

644 **Server-Side Mechanics**

645 Server-side mechanics run exclusively on the game server, which protects critical game logic from
646 player manipulation.

647 Testing server-side mechanics is especially important for multiplayer online games. For example, in
648 player versus player PvP matchmaking, the server ensures fair opponent selection based on player
649 rankings. If this process fails, it could lead to an imbalanced and frustrating gameplay experience.

650 Unlike client-side testing, server-side mechanics are tested using server consoles and database queries
651 rather than UI interactions. Testers often analyze logs and database transactions.

652 Key test types for server-side mechanics include:

- 653 • Functional testing
- 654 • Performance testing
- 655 • Security testing

656 **Client-Server Mechanics**

657 Client-server mechanics combine aspects of both previous types. One part of the logic is processed on
658 the player's device, while another is handled on a remote server. These mechanics continuously
659 interact, exchanging data. In multiplayer games, the server receives data from the client and then
660 distributes the results of actions to all affected players.

661 Data received by the server undergoes mandatory validation. For example, if a player attempts to make
662 a purchase in the in-game store, the server must verify the availability of sufficient in-game currency
663 before completing the transaction. Even if a player modifies the client-side data to gain an unfair
664 advantage, the server will replace it with the correct values.

665 Scenario-based testing can be used to test client-server mechanics. Testers create one or more test
666 cases to cover as many different mechanics as possible.

667 2.1.4 Types of Game Mechanics Failures and Their Possible Causes

668 Regardless of their type, game mechanics are prone to the following types of failures:

- 669 • Mechanics functional failures
- 670 • Mechanics feedback failures
- 671 • Mechanics efficiency failures in specific game environments

672 **Mechanics Functional Failures**

673 This type of failure is directly related to the functional suitability of game mechanics. As with non-gaming
674 software, functional failures occur when a specific feature behaves incorrectly.

675 For example:

- 676 • Weapons do not reload
- 677 • Resources cannot be collected
- 678 • The zoom function does not work when using binoculars

679 These failures typically arise due to defects in the game code.

680 **Mechanics Feedback Failures**

681 Mechanics feedback failures are associated with a lack of proper response from the game when a
682 mechanics is used. For instance, a problematic situation occurs when a player does not understand
683 why the game has ended. This is why game mechanics are often accompanied by visual and sound
684 effects. These could include explosion animations, a countdown timer sound, or even a text notification.

685 Testing evaluates whether a response is present when using the mechanics and evaluates whether its
686 presence or absence is justified. The correctness of the effect itself is verified through other types of
687 testing, such as graphics and sound testing.

688 **Mechanics Efficiency Failures**

689 The third type of defect occurs when a mechanic functions correctly in isolation and provides the
690 expected information to the player, but is not compatible with the other game mechanics with which it
691 interacts. Such a defect may manifest as an inconsistency with comparable mechanics, an imbalance
692 in effectiveness (overpowered or underpowered) during gameplay, or any situation in which the
693 mechanic behaves as intended individually but introduces a system-level defect when integrated into
694 the overall game system.

695 For example, a game designer might add two opponents with different behaviors and characteristics.
696 One is fast and agile but deals low damage, while the other is slow and clumsy but delivers powerful
697 attacks. Both opponents have their strengths and weaknesses and are designed to present similar
698 challenges to the player. However, if most of the game level consists of high platforms that the player's
699 character can climb, the balance changes. The player may safely attack the slow enemy from above,
700 making it less of a threat. Meanwhile, the fast, high-jumping opponent can still pose a significant
701 challenge.

702 Detecting such failures is difficult because predicting these situations and how the mechanics will
703 behave within them is inherently challenging.

704 **2.2 Video Game Mechanics Testing**

705 **2.2.1 Applying Fundamental Approaches to Video Game Mechanics Testing**

706 Game mechanics testing can be performed at various phases of software development, using different
707 test approaches and test types.

708

709 **Concept Phase**

710 At the concept phase, documentation describing the gameplay mechanics and the rules of their
711 operation is reviewed. It is important to understand that some mechanics may be conceived, described,
712 and consequently reviewed at later phase.

713 **Pre-production Phase**

714 During the pre-production phase, only the core mechanics of the game are implemented. The primary
715 tasks of a tester at this phase include verifying mechanics based on the following characteristics:

- 716 • Functional correctness and functional completeness of the mechanics.
- 717 • Functional appropriateness of the mechanics to players.

718 Used Test Approaches:

- 719 • Rapid prototypes: Creating simplified versions of mechanics for quick evaluation of ideas.
- 720 • Game mockups: Testing key mechanics without complex visual components.
- 721 • Iterative approach: Gradually improving mechanics based on collected data and feedback.

722 **Production Phase**

723 During the production phase, implemented mechanics undergo testing to verify they meet the specified
724 requirements.

725 For testing the interaction between different mechanics, exploratory testing and error guessing are
726 particularly effective. In large-scale multiplayer games, the number of possible interactions becomes
727 extremely large and difficult to predict. Exploratory testing allows testers to use their creativity and
728 domain knowledge to discover unexpected interaction defects, while error guessing leverages
729 experience to identify likely problem areas. These experience-based techniques complement scripted
730 testing by uncovering defects that predefined test cases might miss.

731 Manual playtesting with various participant groups is conducted for different purposes:

- 732 • Internal playtesting: Testers and developers play to identify functional defects, balance defects,
733 and usability defects.
- 734 • External playtesting: Real players play while being observed to assess user experience, identify
735 confusion points, and gather feedback.
- 736 • Focus groups: Facilitated discussions with players to gather qualitative feedback on mechanics,
737 appeal, and perceived balance.
- 738 • A/B testing: Players are randomly assigned to different mechanic versions, with quantitative
739 metrics compared to determine effectiveness.

740 When testing with a large number of players, meta mechanics are evaluated, along with the usability
741 and clarity of the game for users.

742 The main objective at this phase is to gather player feedback and assess the impact of implemented
743 mechanics on key metrics such as:

- 744 • Average session length
- 745 • Number of invited friends
- 746 • Average spending per player
- 747 • Most popular in-game items.

748 Used Test Approaches:

- 749 • Mathematical models: Analyzing game balance by using formulas and algorithms.

- 750
- Statistical analysis: Collecting gameplay data (e.g. win/loss ratios, difficulty levels).

751 **Post-Production Phase**

752 After the game is released to the market, testers are responsible for processing and verifying defects
753 in mechanics reported by players, as well as testing new mechanics introduced into the game. Canary
754 testing and A/B testing can be used in this case. Game mechanics testing uses a combination of manual
755 and automated approaches. Automated testing is effective for verifying functional correctness (e.g.
756 damage calculations, state transitions), while manual playtesting is essential for assessing subjective
757 qualities like game feel, balance, and player experience. The appropriate mix depends on the mechanic
758 type and testing objectives.

759

760 **3 Video Game Level Testing - 120 minutes**

761 **Keywords**

762 Defect, failure, geometry testing, playtesting

763 **Video Game Specific Keywords**

764 Block-out, color coding, computer-generated imagery (CGI), game character, game design, game
765 engine, gray box, level editor, setting, structural geometry, terrain, texture, trigger, video game level

766 **Learning Objectives for Chapter 3**

767 **3.1 Principles and Concepts of Level Design in Video Games**

768 GaMe-3.1.1 (K2) Distinguish the video game levels components

769 GaMe-3.1.2 (K2) Give examples of failures in video game levels.

770 **3.2 Test Types in Video Game Levels Testing**

771 GaMe-3.2.1 (K2) Summarize geometry testing.

772 GaMe-3.2.2 (K2) Summarize the playtesting.

773 **3.3 Executing Video Game Level Testing**

774 GaMe-3.3.1 (K3) Apply level testing.

775

776 **3.1 Principles and Concepts of Level Design in Video Games**

777 **3.1.1 The Term "Level" and Its Specifics**

778 A video game level represents a defined environment or scenario in which the player interacts with
779 game mechanics to achieve one or more gameplay objectives. Depending on the game genre and
780 design, a level may range from a simple, self-contained space to a large, complex environment
781 supporting multiple activities and player behaviors.

782 Levels in games consist of multiple components combined into a cohesive whole. The primary
783 components include:

- 784 • Structural geometry
- 785 • Game Environment Objects
- 786 • Textures
- 787 • Lighting, Reflections, and Shadows
- 788 • Sound Effects
- 789 • Gameplay Functionality

790 **Structural Geometry**

791 Structural geometry, or the foundational geometry of game levels, is one of the most crucial components
792 that define any in-game surface. It establishes the terrain on which game characters can move.
793 Structural geometry also limits the playable area of a level. If a game level has clearly designated entry
794 and exit points, structural geometry can guide the player toward the level's completion. Depending on
795 the game's setting, the components forming a game level may vary.

796 **Game Environment Objects**

797 Game environment objects are models or groups of models used to create a more realistic depiction of
798 the necessary game space.

799 Static objects complement and define the terrain or structural geometry of a level. Dynamic objects are
800 interactable elements that are used either directly within gameplay mechanics or to support immersion
801 and the overall player experience.

802 **Textures**

803 All game level objects are covered with textures that simulate different surfaces and materials. These
804 textures can represent components such as walls, floors, terrain, and materials like wood, metal, and
805 stone. They may also include details like footprints, scratches, or dirt. The level of detail and realism in
806 textures significantly influences the player's immersion in the game world.

807 **Lighting, Reflections, and Shadows**

808 The use of different light sources helps create the desired atmosphere in a game level, guide the
809 player's movement, draw attention to specific areas, or hide them from enemies.

810 **Sound Effects**

811 Background music and sound effects accompany various actions and events within the game.

812 **Gameplay Functionality**

813 Gameplay functionality consists of various settings that organize the gaming process on a particular
814 level. These settings include:

- 815 • Settings related to level progression:
 - 816 ○ Player and enemy spawn points
 - 817 ○ Win and loss conditions
 - 818 ○ Automatic save points
 - 819 ○ Scripted events triggering specific game actions or sequences (e.g., an airplane flying
820 overhead every two minutes, a cutscene playing when a character enters a building, or
821 an alarm activating upon entering a restricted area)
- 822 • Settings for interactive objects such as doors, vehicles, destructible objects, and traps
- 823 • Physical boundaries preventing the character from leaving the level or getting stuck in its
824 geometry

825 Each component of a game level may contain defects. Defects affecting lighting, shadows, and
826 reflections are typically identified through graphical system testing, while defects in sound content are
827 detected during audio testing. These aspects are covered in the respective sections of this syllabus.

828 **3.1.2 Types of Video Game Level Failures**

829 There are several types of failures that commonly occur during the creation of video game levels. These
830 failures can be difficult to detect and may remain unnoticed for some time. In most cases, the best way
831 to identify level failures is through testing by experienced players who can recognize anomalies and
832 report defects.

833 A significant portion of failures found in video game levels can be grouped into several common
834 categories, each affecting gameplay, navigation, or player perception in different ways.

835 1. Navigational and structural failures affect player movement and access within the level and
836 include failures such as incorrect object scaling, stuck points, map holes, inaccessible areas,
837 and invisible walls.

838 **Incorrect Object Scaling**

839 This type of failure includes objects on the map that are improperly scaled and may be used by players
840 as cover. For example, a player may be unable to take cover behind a wall or a crate even when
841 crouching. Such failures can alter the entire gameplay experience and/or provide an unfair advantage
842 to certain players.

843 **Stuck Points**

844 A game character may become stuck in the level's geometry, unable to escape, or may enter restricted
845 areas. Identifying such failures is especially important in multiplayer games, where level design defects
846 can provide an unfair advantage to certain players or teams.

847 Stuck-related failures can be caused by various factors. One common cause is level reconstruction,
848 which may disrupt the navigation mesh or its automatic generation. For example, if models are
849 repositioned during level reconstruction but the navigation mesh remains unchanged, non-player
850 characters (NPCs) may attempt to move through walls or other objects they cannot bypass, resulting in
851 them getting stuck—a situation commonly referred to as a stuck point.

852 **Map Holes**

853 Another common failure is map holes. These failures result from incorrect level geometry, where parts
854 of the level model (such as walls and floors) do not fit tightly together. As a result, a character may fall
855 through the map, entering an area with no support, causing them to endlessly fall into the void due to
856 gravity.

857 **Inaccessible Areas**

858 In video games, especially in multiplayer maps, there may be intentional locations where a player gains
859 a tactical advantage over others. For example, a machine gunner positioned on a hill may have a wide
860 firing sector. While the first player to reach such a position gains an advantage over opponents, this is
861 a deliberate design choice and not a failure.

862 However, there are cases where such areas are unintentionally created due to level design defects.
863 The developer may have intended a specific area to be inaccessible, but a certain character, utilizing
864 unique abilities (such as higher jumps), may manage to reach it. As a result, the player may gain an
865 unfair gameplay advantage over opponents, such as being undetectable or immune to attacks.

866 **Invisible Walls**

867 These failures are usually the result of an incorrect object modeling. They often occur when the visible
868 model of an object does not match its collision model—the physical model that defines the object's
869 tangible boundaries.

870 2. Visual and presentation failures impact immersion and clarity and include floating objects,
871 incorrect color coding, texture seaming, and failures related to the recognizability of real
872 (historical) prototypes.

873 **Incorrect Color Coding**

874 Objects encountered by the player during gameplay have different properties. Some of them serve as
875 decorations, while others are interactive. Color coding of objects with different properties is an essential
876 aspect of level design in video games. Therefore, once a specific color coding is established, it should

877 remain consistent throughout the game. Similarly, breakable crates, doors that can be opened,
878 climbable rocks, and ledges should be visually distinct in shape and/or color from other objects.

879 If color coding is inconsistent, players may waste time figuring out which objects are interactive and
880 what actions they need to take to progress.

881 **Texture Seaming**

882 Such failures are typically noticeable on large maps and locations where different surface textures, such
883 as sand and grass or sand and water, blend together. The transition between textures may not always
884 appear seamless and natural.

885 **Real (Historical) Prototypes Recognizability**

886 In games set in worlds that mimic reality, it is crucial to maintain resemblance to recognizable objects.
887 These may include architectural landmarks, landscapes, and geographical locations. Historical
888 accuracy must also be preserved, as its absence can disrupt player immersion.

889 3. Gameplay and design-related failures influence player progression and experience and include
890 unnecessary gameplay complications, artificial restrictions and constraints, narrative
891 inconsistencies, and failures related to destructible environments.

892 **Floating Objects**

893 During level development, multiple specialists may work on the same location simultaneously. Some of
894 them may add or remove objects, while others modify the terrain geometry by raising or lowering
895 sections of the map. As a result, failures may occur where previously grounded objects become
896 suspended in mid-air.

897 **Unnecessary Gameplay Complications**

898 These failures arise when there is a lack of clear objective markers or when required actions to progress
899 through a level are not intuitive. For example, a player may defeat all enemies in the accessible area
900 but remain uncertain about the next step needed to advance.

901 **Artificial Restrictions and Constraints**

902 These failures do not impact gameplay balance or provide unfair advantages but negatively affect
903 immersion and player perception. For example, a player may encounter a waist-high fence blocking
904 their path but find it impossible to climb over. While the level designer may have intended this area to
905 be inaccessible, from the player's perspective, such a limitation feels like an oversight that disrupts
906 immersion. Similarly, a player may perceive it as a defect when a character is trapped behind a metal
907 grate, unable to escape, despite the gaps between the bars being wide enough to pass through.

908 **Level Balance**

909 Many failures arise from improper level balance adjustments. For example, overly difficult enemy
910 characters at a given phase of the game may make level completion impossible. Such failures are often
911 identified during AI testing. In multiplayer games, where multiple teams compete to reach a designated
912 point on the map, level balance failures may include unfair starting conditions that give one team an
913 advantage over another.

914 **Narrative Inconsistencies**

915 Level design failures disrupt the overall storytelling style or game narrative. While these failures have
916 minimal impact on core gameplay mechanics, they negatively affect player immersion and the overall
917 perception of the game. For example, if a character arrives in a city that, according to the story, was

918 devastated by a nuclear bombing, the presence of undamaged buildings would create a narrative
919 dissonance.

920 **Destructible Environment Failures**

921 If the game environment is non-interactive — meaning the player cannot interact with it — the level of
922 realism remains low, even if environmental objects are accurately modeled. The concept of a
923 "destructible environment" encompasses various components such as terrain (referred to as
924 "deformable terrain"), vegetation, interiors, buildings, and other structures. Special attention is given to
925 destructible environments in games where they play a crucial role in gameplay or even become part of
926 core game mechanics.

927 A destructible environment can also be integrated into the storyline. For example, the destruction of a
928 wall might block the player's path, forcing them to find an alternative route pre-programmed by the
929 developers.

930 4. Performance-related failures occur when level design or content causes unacceptable
931 degradation in frame rate, loading times, or system responsiveness. Common causes include
932 overly dense geometry, duplicated or overlapping meshes, excessive use of high-resolution
933 textures, inefficient lighting setups, or poorly optimized environmental effects

934 **3.2 Test Types in Video Game Levels Testing**

935 Considering the characteristics and development sequence of game levels—from geometric
936 prototyping to the final refinement of details and objects—it is logical to distinguish two levels of video
937 game level testing:

- 938 • Geometry Testing (including texturing)
- 939 • Playtesting

940 Testing at these levels does not consist of isolated activities. The division is based on the need to focus
941 on identifying defects with a common root cause. This approach not only simplifies defect detection but
942 also ensures that defects are properly addressed for subsequent fixes.

943 **3.2.1 Geometry Testing**

944 Testing of game levels begins at the earliest phases of their creation. The test objects, the nature of
945 tests and the order of execution, as well as the responsibilities of various specialists, depend on the
946 current phase of level development.

947 **Block-Out/Gray Box**

948 Based on a previously developed sketch, in 3D video games a three-dimensional model of the future
949 game level is created using special objects. Only components that directly impact gameplay are used:
950 gray cubes, spheres, cylinders, and planes that schematically form the layout of the game level.

951 In this phase, not all game mechanics may be fully implemented, requiring the level designer to
952 gradually integrate them while adapting the level to the constantly evolving development conditions.

953 To create a prototype with proper proportions and scaling, the level designer collaborates with the game
954 designer to define a set of measurements and parameters for the game character. These
955 measurements directly influence the size of objects and environmental components within the level.

956 Once the level designer has determined the appropriate scale and size for the game level and its
957 objects, additional functionality is configured to support gameplay. This includes spawning points for
958 characters, scripted events, and other interactive components.

959 **Final Version**

960 The level designer integrates the software components created by the entire development team into a
961 cohesive whole. At this point, decorative details are added to the game level, and objects begin to
962 resemble real-world counterparts not only in appearance but also in function and placement. During the
963 development of the level geometry, it is essential to conduct functional and visual validation tests
964 focusing on:

- 965 • Object proportions and dimensions within the game level, ensuring consistency with game
966 design documents and visual expectations.
- 967 • Level boundaries, to verify they behave as expected (e.g., collision boundaries, invisible walls,
968 playable area limits).
- 969 • Texturing fidelity, which includes correct mapping, alignment, and resolution consistency
970 across surfaces.

971 These tests help detect visual inconsistencies, clipping defects, and environmental design defects, such
972 as map holes, contributing to both playability and immersion quality.

973 **3.2.2 Playtesting**

974 The test objective of this type of level testing is to identify defects that are not directly related to its
975 geometry. This process involves evaluating all aspects that enhance player immersion and improve
976 overall gameplay convenience.

977 Key focus areas of such testing include:

- 978 • Overall level balance
- 979 • Absence of inaccessible areas or logical justification for their inaccessibility
- 980 • Clear and intuitive navigation and level orientation
- 981 • Elimination of unnecessary complexity and artificial constraints
- 982 • Recognizability of objects with real (historical) prototypes
- 983 • Absence of invisible and floating objects
- 984 • Environmental destructibility (if applicable)

985 Since game levels are typically vast spaces, designing detailed test cases to verify all of the above can
986 be challenging or even impractical. A crowdtesting can be an appropriate test approach in this case.

987 In addition, testers should pay attention to how the level supports different gameplay styles — for
988 example, stealth versus aggressive approaches. The placement of cover, interactive component, and
989 resources should support these styles without giving unfair advantage to one. Sound design and
990 ambient cues also play a significant role in level perception and should be validated as part of the overall
991 experience. Player feedback gathered during exploratory testing is especially valuable for identifying
992 subtle defects that formal test plans might miss.

993 **3.3 Executing Video Game Level Testing**

994 **3.3.1 Applying Fundamental Approaches to Level Testing**

995 Earlier, different phases of level creation were discussed, from geometric prototyping to the final
996 version. From a testing perspective, the varying degrees of completeness of a level determine which
997 test approaches are appropriate and what aspects of the level can be meaningfully evaluated at each
998 phase. Testing activities should therefore be selected and applied based on what is implemented and
999 stable, rather than treating all phases as equally testable.

1000 While playtesting is performed throughout level development, its focus evolves as the level matures.
1001 Early playtesting emphasizes basic layout, navigation, and mechanical feasibility, whereas later
1002 playtesting evaluates gameplay flow, balance, performance, and overall player experience.

1003 **Testing the Prototype of a Video Game Level**

1004 Since requirements are not yet finalized at the prototyping phase, exploratory testing and other
1005 experience-based test techniques are considered the most effective.

1006 Initial playtesting are conducted on the created prototype to assess the gameplay on this map.
1007 Participants in these early playtesting may include testers from the game development team as well as
1008 external testers. By gathering feedback from playtest participants, developers can evaluate successful
1009 and unsuccessful design choices and make necessary adjustments, relocations, or removals of objects
1010 within the prototype.

1011 Additionally, playtesting of the prototype evaluates game mechanics and their interaction with object
1012 placement and size within the level layout.

1013 For example, cover components on the game level must be large enough for the character to hide
1014 behind them, surface gaps must be appropriately sized for the character to jump across, and ledges
1015 must be accessible for climbing without the character hitting their head on low ceilings.

1016 Furthermore, testing evaluates if the level layout creates the intended game difficulty and provides the
1017 right gameplay experience. For instance, if a section of the game is designed to feature a firefight with
1018 multiple enemies, the level must contain a sufficient number of cover points. Conversely, if the level
1019 includes a battle against a single powerful and resilient enemy (a boss fight), the playable area should
1020 be constrained and cleared of unnecessary objects to enhance the intended challenge.

1021 **Prototype Geometry Testing**

1022 In this phase, testing validates the actual results obtained by the artistic rendering of the game level
1023 that could interfere with gameplay.

1024 Examples of such failures include a temporary 3D model of a rock blocking the level's passage or a
1025 tree canopy obstructing the player's view.

1026 Additionally, tests are performed to evaluate that there are no gaps between essential level
1027 components, such as the "floor" and other objects, to prevent unintended voids in the map.

1028 To identify these anomalies, playtesting is conducted again, focusing on evaluating game mechanics
1029 concerning the size and placement of objects according to required metrics and ensuring the
1030 correctness of gameplay.

1031 Simultaneously, additional testing may be carried out for object appearance, lighting, game
1032 environment, and other visual aspects.

1033 **Final Version Testing**

1034 As part of playtesting, testers verify the presence of collision models for all game objects, ensuring that
1035 physical and visual object models align correctly.

1036 In this phase, performance testing and compatibility testing is conducted. The functional suitability of
1037 the game level is tested on various devices, frame rate measurements are taken, and the quality of
1038 object rendering at different distances, lighting, and shadows is verified.

1039 Collision model testing can be performed as follows: the tester moves the game character toward an
1040 object, such as a large rock, and attempts to "pass through" it from different angles. If the rock's collision

- 1041 model is properly configured, the character's body will only visually touch the rock without passing
1042 through or becoming embedded in it.
- 1043 In this phase, essential tests include verifying the correctness of object texturing (ensuring no visible
1044 "seams"), confirming the alignment of objects with real (historical) prototypes.
- 1045

1046 **4 Graphics Testing - 95 minutes**

1047 **Keywords**

1048 Defect, failure, playtesting

1049 **Video Game Specific Keywords**

1050 3D model, animation, clipping, collision, hitbox, level of detail (LoD), loot, mapping, occlusion, scene
1051 lighting, speedrunning, texture, video game level, visual effect (VFX)

1052

1053 **Learning Objectives for Chapter 4**

1054 **4.1 Principles and Concepts of Video Game Graphics**

1055 GaMe-4.1.1 (K1) Recall the types of video game graphic components

1056 GaMe-4.1.2 (K2) Give examples of failures in video game graphics

1057 **4.2 Test Types in Video Game Graphics Testing**

1058 GaMe-4.2.1 (K1) Recall artistic testing

1059 GaMe-4.2.2 (K1) Recall technical testing of graphics.

1060 GaMe-4.2.3 (K1) Recall playtesting.

1061 **4.3 Executing Video Game Graphics Testing**

1062 GaMe-4.3.1 (K3) Apply graphics testing

1063

1064 **4.1 Principles and Concepts of Video Game Graphics**

1065 **4.1.1 Features of Video Game Graphics**

1066 Any gaming product consists of numerous graphical components, collectively enabling users to
1067 visualize the gameplay experience.

1068 **Models**

1069 A model is an object within computer graphics. Based on image creation methods, graphical models
1070 can be categorized as follows:

- 1071 • Two-dimensional computer graphics (2D)
- 1072 • Three-dimensional computer graphics (3D)
- 1073 • Computer-generated imagery (CGI)

1074 **Model Animation**

1075 It is the process of making 3D characters, creatures, or objects move and behave in a lifelike or stylized
1076 way. It involves applying motion to digital models so they can walk, run, talk, gesture, or interact with
1077 their environment during gameplay.

1078 **Model Collision**

1079 Models within a video game scene interact with one another. Any interaction begins with contact
1080 (collision). To enable a model to interact with its environment and other objects, a collider is created —
1081 an invisible simplified shape attached to the object to calculate collisions with other objects. The collider
1082 can be considered the physical model of the object. While it remains invisible to the player, any collisions
1083 with colliders are processed by the game engine.

1084 However, some objects in the video game may serve purely decorative purposes and may not have
1085 any collision detection at all.

1086 **Model Realism and Authenticity**

1087 In addition to entertainment, video games can often include an educational component. In such cases,
1088 maintaining realism and adherence to real (historical) prototypes in the use of graphical assets becomes
1089 particularly important.

1090 Inaccurate details can not only distort the perception of history but also undermine the developers'
1091 efforts in crafting an authentic atmosphere.

1092 **Model Textures**

1093 In a broader sense, textures can be considered as patterns on the surface of a sculpted object. Using
1094 textures allows for the reproduction of fine surface details that would be excessively complex to create
1095 with polygons, such as scars on the skin, wrinkles on clothing, small rocks, and other fine details on
1096 walls and terrain surfaces.

1097 **Visual Effects (VFX)**

1098 Visual effects can be categorized into two main types:

- 1099 • Gameplay Effects
- 1100 • Environmental Effects

1101 VFX consist of a combination of animations, lighting effects, particles, and other graphical components
1102 that make the video game world more realistic, dynamic, and engaging.

1103 **Scene Lighting, Reflections, and Shadows**

1104 Scene lighting is used to create different moods for the player, alter their perception of in-game events,
1105 and encourage specific behaviors. Thus, many lighting techniques used in visual arts, cinema, and
1106 architecture are also applied in video games to enhance the aesthetics of virtual spaces and improve
1107 player immersion.

1108 4.1.2 Types of Failures in Video Game Graphics

1109 The classification of failures related to the graphical subsystem of a video game is based on and directly
1110 linked to the areas of this subsystem mentioned in the previous section.

1111 **Types of Software Failures in Video Game Graphics**

- 1112 • Model failures
- 1113 • Texture mapping failures
- 1114 • Collision failures
- 1115 • Hitbox failures
- 1116 • Realism and authenticity failures
- 1117 • Animation failures
- 1118 • Animation transition failures
- 1119 • Dismemberment gore system failures

- 1120 • Camera shake failures

1121 **Model Failures**

1122 To better understand the root causes of defects related to failures that arise in the graphical subsystem
1123 of a game, as well as the verification process, it is essential to first examine how graphical objects are
1124 created and how they interact with each other.

1125 When working with 2D models, the verification process in the early phases is relatively straightforward:
1126 the object should be evaluated for accuracy (if it has a real-world prototype) and how different users
1127 perceive it during localization. However, 3D models introduce additional complexities.

1128 A 3D model is composed of polygons—multi-sided shapes whose corner points have precise
1129 coordinates in three-dimensional space and are connected by lines. The higher the level of detail, the
1130 greater the number of polygons required for modeling.

1131 It is important to recognize that calculating the movement of a large number of points during model
1132 animation (especially when many such models are present in a game scene) places a significant load
1133 on the hardware of the gaming platform. This, in turn, substantially increases the performance
1134 requirements of the graphics processor (GPU).

1135 As a result, high polygon counts reduce the potential player base for the game by limiting accessibility
1136 to users with lower-end hardware. This can significantly impact the game's commercial success.

1137 Thus, one of the key tests in technical testing is verifying that the model is optimized for use within the
1138 game engine and will not cause performance failures. These tests are conducted regularly throughout
1139 development to enable continued optimization.

1140 **Texture Mapping Failures**

1141 In the phase of applying textures to the created model, various failures may arise, including:

- 1142 • Incorrect texture scaling, causing distortion of the image on the surface of the modeled object.
- 1143 • Visible seams between textures covering different parts of the model.
- 1144 • Selection of low-quality, low-resolution textures, reducing the visual appeal.
- 1145 • Incorrect transparency settings, as well as misconfigured reflection and refraction parameters.
- 1146 • Repetitive texture tiling on a single surface, significantly reducing the model's realism.

1147 The primary aspect of texture testing is verifying their presence on visible objects, ensuring correct
1148 display and uniformity. If the video game uses high-resolution textures, there should be no low-quality
1149 textures disrupting the overall visual consistency.

1150 **Collision Failures**

1151 All model interactions occur through collisions. Most models in a video game include a collider, and the
1152 size and configuration of this collider play a critical role in the emergence of collision failures.

1153 If an object's collider is much smaller than its visible model, the character may "clip through the texture"
1154 or walk through it. Strictly speaking, the phrase "clip through the texture" (for example, when a character
1155 can pass through another object) is technically incorrect but accurately describes what the player sees
1156 when their character is partially or fully submerged inside another object.

1157 These failures are especially critical in multiplayer games, as they can provide an unfair advantage to
1158 certain players. This is why testers must pay special attention to objects on PvP maps to evaluate if
1159 there is a fair gameplay.

1160 If a collider is larger than the visible model, situations can arise where the character runs into an invisible
1161 wall or stands on an invisible platform. These failures are often exploited for speedrunning, where
1162 players deliberately search for areas where developers forgot to properly adjust colliders. In some
1163 cases, this allows players to bypass large sections of a level and significantly reduce completion time.

1164 **Hitbox Failures**

1165 Character interaction in games often occurs aggressively, meaning they involve damage infliction.
1166 Therefore, a hitbox is included in the collider model.

1167 A hitbox is a part of the physical model, which registers a collision with a programmed object (e.g., a
1168 bullet, spear, sword, or fist) as damage dealt. In other words, if an object designed to deal damage
1169 collides with the hitbox of another object's physical model (collider), a certain number of health points
1170 is subtracted from the second object's health variable. To maintain realism, a model—such as that of a
1171 human-like creature—may have multiple distinct hitboxes to simulate varying levels of damage.
1172 Therefore, the tester's task is to verify the correct placement of hitboxes and the accuracy of damage
1173 application when interacting with them, as defects in this area can lead to significant player frustration
1174 and dissatisfaction.

1175 **Realism and Authenticity Failures**

1176 Failures related to real-prototype accuracy occur when the visual or behavioral attributes of gameplay
1177 objects, based on real characters, items, or events, are significantly distorted. These inconsistencies
1178 may include incorrect proportions, colors, animations, or details that compromise historical or cultural
1179 authenticity. Such defects negatively impact the realism and immersion of the game world, particularly
1180 in simulation or historically themed games, and may lead to dissatisfaction among players and public
1181 criticism.

1182 **Animation Failures**

1183 Failures related to model animation can arise due to incorrect operation of the skeletal joint system,
1184 which may manifest as chaotic or unnatural limb movements.

1185 Similarly, failures related to facial animation may occur, appearing as a complete lack of animation, an
1186 unnatural transition from one facial expression to another or a facial animation and the contextual
1187 situation mismatch.

1188 A specific case of animation failures is the lack of lip sync, where lip shapes corresponding to specific
1189 phonemes are missing, incorrect sequences of lip shapes are used or animation is entirely absent.

1190 **Animation Transition Failures**

1191 Since a model can use different animations for performing various actions (idle, walking, running,
1192 jumping), various failures may occur.

1193 A common failure related to animation transitions is the so-called "sliding" effect. This occurs when a
1194 character abruptly changes movement direction, such as suddenly turning and running to the side after
1195 moving forward. Such errors are especially noticeable in third-person games and often arise due to an
1196 insufficient number of animations for smoothly displaying different speeds and movement directions.

1197 **Dismemberment Gore System Failures**

1198 The effect of dismemberment is implemented by developers to simulate the detachment of individual
1199 limbs or the complete dismemberment of a character's body.

1200 To achieve this effect, the model must be pre-prepared by "dividing" it into smaller parts. The effect is
1201 triggered when multiple conditions are met simultaneously:

- 1202 1. The character must have less than a specified amount of health points (HP).
1203 2. The damage must be inflicted by a specific type of weapon.
1204 3. A shot from this weapon must hit a specific area of the character's model.

1205 When the specified conditions are met, the effect is triggered, resulting in the detachment of a part of
1206 the model or its "explosion," scattering fragments and adding special effects such as blood and other
1207 details.

1208 Failures that may occur include unnatural detachment of limbs, remaining parts of the model staying in
1209 place after the "explosion" and/or absence of the intended special effects.

1210 **Camera Shake Failures**

1211 The effect is used in various gameplay situations to enhance realism (e.g., during walking, jumping,
1212 shooting) or to inform the player about a specific in-game event (e.g., taking damage). The essence of
1213 this effect is that the camera "shakes," shifting relative to its initial position with a set intensity for a
1214 specific duration. This effect is especially common in console games, where camera vibration is
1215 accompanied by controller vibration. A failure would be the absence of this effect in the situations
1216 described above.

1217 **Types of Hardware Failures in Video Game Graphics**

- 1218 • Screen tearing
1219 • Ghosting
1220 • Flickering
1221 • Level of detail (LoD) failures
1222 • Collision failures/Clipping
1223 • Occlusion culling failures
1224 • Z-Fighting
1225 • Missing textures
1226 • Visual artifacts (graphical "glitches")
1227 • Failures in visual effects
1228 • Lighting, reflections, and shadows failures

1229 **Screen Tearing**

1230 This failure is not directly related to the graphical components of the game but arises due to hardware
1231 characteristics. However, it has a noticeable visual manifestation in the form of scene distortions. The
1232 failure occurs because monitors have a specific refresh rate (e.g., 60 Hz, 144 Hz, 165 Hz, 240 Hz),
1233 which determines the number of image updates per second. If the graphics card generates more frames
1234 than the monitor can display, frames start overlapping, creating a "tearing" effect.

1235 **Ghosting**

1236 It is a failure, which appears as the overlay of several displaced images of an object on the screen. It
1237 occurs when the previous image remains visible simultaneously with the current one during the
1238 transition to the next frame, creating a blurred or "ghosting" effect.

1239 The main causes of this phenomenon are related to the screen refresh rate and response time. The
1240 refresh rate determines how often the monitor updates the displayed image, while the response time is
1241 the period during which the processor transmits a new image to the display, replacing the old one. If
1242 the response time is too short, the previous image does not fully disappear, leading to the occurrence
1243 of the ghosting.

1244 Other causes of ghosting include:

- 1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
- Temporal anti-aliasing algorithms, which rely on data from previous frames and, when improperly tuned or when motion vectors are insufficient, lead to the accumulation of trailing artifacts.
 - Upscaling and temporal upscalers, where motion reconstruction errors or mismatches between motion vectors and actual geometry result in residual artifacts.
 - Low or unstable frame rates, which amplify frame-to-frame differences and cause temporal filters to behave more aggressively.
 - Motion vector defects (missing, inaccurate, or mismatched with animation, skinning, or VFX), causing the engine to incorrectly “predict” pixel movement.
 - Post-processing effects (motion blur, bloom, volumetric effects), which, when combined with temporal filtering, further intensify trailing and ghosting artifacts.

1256 **Flickering**

1257 This often refers to a situation where an object (such as a texture or particles) periodically blinks, flickers,
1258 disappears, and reappears. Such behavior may be related to response time settings.

1259 **Level of Detail (LoD) Failures**

1260 LoD models are divided into levels (for example, LoD0 — the most detailed model, LoD1 — a slightly
1261 simplified one, and so on up to LoD3 or LoD4). The game dynamically selects the appropriate model
1262 level depending on:

- 1263
1264
1265
1266
- the distance of the object from the camera;
 - the current frame rate (FPS);
 - the speed of object movement;
 - the number of simultaneously visible objects in the scene.

1267 Despite the effectiveness of the technology, its implementation may be accompanied by various
1268 failures. For example:

1269 Abrupt "level popping" occurs when the transition between levels of detail is too noticeable.

1270 On low-performance systems, the high-detail model may not load in time, and the player observes a
1271 rough, simplified version of the object even at close range.

1272 Simplified LoD models may sometimes fail to preserve the correct shape or proportions of the original,
1273 which can affect the scene's aesthetics or confuse the player.

1274 **Collision failures/Clipping**

1275 Clipping occurs when an object that should be invisible (for example, one that is outside the field of view
1276 or hidden behind an opaque obstacle) becomes visible.

1277 **Occlusion Culling Failures**

1278 Failures caused by incorrect occlusion culling algorithms result in the display of scene components that
1279 should be hidden.

1280 If one opaque object partially covers another; the obscured part of the second object should not be
1281 visible. Ideally, such areas are excluded from the rendering process to optimize computational
1282 resources. However, if the graphics processor failures, the user may see object components that should
1283 remain hidden.

1284 **Z-Fighting**

1285 This rendering failure occurs when two primitives (basic geometric shapes used as a starting building
1286 block for creating more complex 3D models) alternately overlap each other, creating a flickering effect
1287 on the screen.

1288 The reason for this failure is that both primitives are positioned so closely to the camera that the graphics
1289 processor cannot determine which one should be displayed in a specific pixel. This happens due to
1290 uncertainty in the operation of the Z-buffer, which manages depth, as the coordinates of the primitives
1291 nearly coincide.

1292 When the scene changes, the defect manifests as flickering or graphical noise, as data from one of the
1293 primitives is alternately used to render the screen pixels.

1294 **Missing Textures**

1295 This failure appears as placeholders in place of missing textures. The game's code contains information
1296 about textures and their locations, but the textures themselves are either absent or the graphics
1297 processor fails to load them from storage in time to display them on the screen.

1298 **Visual Artifacts (Graphical "Glitches")**

1299 These failures are difficult to classify within the previously described categories, and sometimes their
1300 exact cause is hard to determine. Such "glitches" are most often related to the unstable operation of
1301 the graphics processor, its drivers, or other hardware and software components.

1302 An example of such failures is "wandering pixels", which are artifacts that appear around objects and
1303 disappear when the viewing angle changes.

1304 **Failures in Visual Effects**

1305 Failures in visual effects may include:

- 1306 • Missing effects where they are necessary (e.g., no ripples on a pond's surface in windy
1307 weather).
- 1308 • Violation of visual effect implementation principles, including realism (e.g., a grenade explosion
1309 consisting only of smoke).
- 1310 • Excessive or insufficient speed of effect display (e.g., flames in a fireplace moving too fast).
- 1311 • Effect duration being too short or too long (e.g., a gunshot muzzle flash persisting after the shot
1312 has been fired).
- 1313 • Breaking the laws of physics, optics, in effect implementation (e.g., extremely dense fog
1314 completely obscuring the game scene).

1315 It is evident that testers, not being a specialist in visual effect creation, cannot fully evaluate them.
1316 However, they can assess effects based on the principles of their implementation, identify where an
1317 effect is missing but should be present, detect its improper placement or timing, notice distortions, overly
1318 short or long durations, and unrealistic environmental interactions. Additionally, a tester can observe if
1319 an effect becomes repetitive and starts to annoy the player over time. These are the key aspects that
1320 should be identified during special effects testing as part of game graphics verification.

1321 **Lighting, Reflections, and Shadows Failures**

1322 Proper lighting setup in video games, just like in cinema or visual arts, serves as an essential tool for
1323 achieving specific objectives: creating atmosphere, emphasizing object volumes, and simplifying
1324 navigation through the game level by harmoniously combining light and color.

1325 Lighting helps highlight key points, such as save points or resource refill stations, indicate loot on the
1326 map, or mark cover spots where the player can remain undetected.

1327 Lighting can be natural or artificial, directional, reflected, or refracted, and can also vary in intensity.
1328 Common lighting failures are typically caused by improper use of light sources, incorrect simulation of
1329 optical effects, or shadows cast by objects.

1330 Lighting failures also include disrupted auto-exposure that is, the imitation of how human vision adapts
1331 when moving from darkness into a lit environment.

1332 Another lighting failure is the improper use or handling of glare. Glare is used to add realism to a scene's
1333 lighting and to imitate very bright light sources.

1334 4.2 Test Types in Video Game Graphics Testing

1335 The process of creating graphic content, including two-dimensional or three-dimensional models,
1336 special effects (VFX), implementation of lighting for game scenes, reflection systems, and shadows, is
1337 highly complex and consists of several key phases. It is essential to remember that testing should begin
1338 as early as possible. Essentially, there are three test types for testing video game:

- 1339 • Artistic Testing (verification of visual, including aesthetic aspects)
- 1340 • Technical Testing (verification of technical parameters and requirements)
- 1341 • Playtesting (includes verification of animation and object interactions)

1342 It is important to understand that these are not separate independent activities within the graphics test
1343 process. The distinction between these test types is primarily aimed at more accurately identifying the
1344 root causes of defects and the specialists responsible for fixing them in the future. Additionally, each
1345 test type employs its own methods and approaches, which differentiate test activities from one another.
1346 Different specialists may be involved at each phase, ranging from artists to regular users.

1347 4.2.1 Video Game Graphic Artistic Testing

1348 During the creation process, a graphical object undergoes numerous phases and tests. In large
1349 projects, specialists involved in developing graphic assets are often included in the test process.

1350 Testing at this level is called artistic testing. The specialist's task is to evaluate the visual and aesthetic
1351 aspects of a graphical object, specifically:

- 1352 • Object geometry
- 1353 • Texture quality
- 1354 • Compliance of models with their real (historical) prototype
- 1355 • Model animation
- 1356 • Visual effects
- 1357 • Scene lighting, object reflections, shadows
- 1358 • Consistency of visual components (ensuring that textures, models, and effects appear as part
1359 of a cohesive world)

1360 The goal is to enhance the visual quality of models before exporting them to the game engine. Extensive
1361 experience and access to development tools allow artists, 3D modelers, and animators to identify
1362 defects much faster and more efficiently.

1363 Artistic testing is a complex task that can be performed at various production phases, from creating
1364 basic geometry and textures to exporting the model to the engine and placing it on the map. For artistic
1365 testing, testers often do not require specialized tools or graphical content editors — it can be conducted
1366 directly within the game.

1367 4.2.2 Video Game Graphics Technical Testing

1368 The creation of graphics involves compliance with a large number of technical requirements, ranging
1369 from the polygon count of a 3D model to complex parameters for calculating object reflections.
1370 Technical testing is aimed at detecting defects related to:

- 1371 • Graphic file formats
- 1372 • Presence of necessary graphic files in the appropriate directories
- 1373 • Integrity of these files
- 1374 • Collision model size
- 1375 • Correct placement of hitboxes
- 1376 • Compliance of the model with the game engine's requirements regarding polygon count
- 1377 limitations
- 1378 • Distance settings for model switching
- 1379 • Verification of timely switching between destruction models and other technical aspects

1380 The goal of technical testing is to verify that all technical parameters of graphical objects are met for
1381 their correct display in game scenes while also optimizing hardware performance on the gaming
1382 platform.

1383 Graphics testing occurs in parallel with artistic testing and may require additional tools. Some tests can
1384 be conducted directly within the game engine or in a 3D editor.

1385 4.2.3 Video Game Graphics Playtesting

1386 At the gameplay level, the tester's task is to verify the interaction of graphical objects in the game scene
1387 concerning their impact on gameplay. This may involve evaluating whether a model's animation aligns
1388 with the in-game situation or testing whether game object settings comply with the requirements of the
1389 game mode.

1390 It is fair to say that playtesting primarily identifies defects affecting the overall player experience. In other
1391 words, it determines how well the graphical components of a video game enhance immersion without
1392 disrupting gameplay.

1393 The main failures detected at the gameplay level include:

- 1394 • Desynchronization of animations with game events
- 1395 • Incorrect rendering of effects during rapid camera movement
- 1396 • Unnatural interaction of visual effects with the environment
- 1397 • Inconsistency of graphics with the game's artistic style
- 1398 • Graphical failures when switching between different quality settings (low, medium, high)
- 1399 • Lack of emphasis on important game components, causing them to blend into the background
- 1400 • Excessive graphical components that distract from the gameplay
- 1401 • Flickering textures or "artifacts"
- 1402 • Objects disappearing from view at certain camera angles
- 1403 • Annoying or overly intrusive visual effects
- 1404 • Excessive hardware load on the gaming platform

1405 4.3 Executing Video Game Graphics Testing

1406 The approach to testing video game graphic content does not involve radically different test techniques
1407 from those used in software testing. Since a graphic object is a system component, it is logically justified
1408 to apply a methodology based on verifying such components at different phases of their creation and

1409 integration. The primary difference in testing lies in the level of isolation of the graphical subsystem
1410 object. In this case, testing is performed sequentially as follows:

- 1411 • Component (Unit) Testing
- 1412 • Integration Testing
- 1413 • System Testing (often including certification/compliance testing)

1414 At the unit test level, objects are examined during their creation. Here, graphical objects undergo several
1415 key verifications:

1416 **1. Model Testing**

- 1417 • Geometry Testing – verifies that 3D models do not contain overlapping polygons or
1418 unnecessary components.
- 1419 • Polygon Optimization – evaluates whether the model meets the specified polygon limits
1420 for a particular game engine.
- 1421 • UV Mapping Testing – evaluates the correctness of the unwrapping for subsequent
1422 texture application.

1423 **2. Texture Testing**

- 1424 • Texture Quality – verifies the absence of artifacts, blurriness, or incorrect resolution.
- 1425 • Compatibility – verifies that textures are correctly displayed in the game engine.
- 1426 • Seams and Joints – evaluates that textures align smoothly without visible seams.

1427 **3. Testing of Real (Historical) Prototypes (where applicable)**

1428 **4. Animation Testing**

- 1429 • Smoothness of movements – verifies the absence of jerky movements, pose
1430 inconsistencies, or other visual failures.
- 1431 • Animation blending – verifies that transitions between animations appear natural.

1432 **5. Special Effects (VFX) Testing**

- 1433 • Effect Physics Compliance – verifies whether the effect adheres to the laws of physics
1434 (or the game's rules).
- 1435 • Effect Duration – verifies that the effect duration is appropriate.

1436 It is important to remember that testers may encounter certain challenges in this phase, such as:

- 1437 • Some failures may only appear in motion or at specific viewing angles.
- 1438 • Visualization results may vary depending on the graphics card or display.
- 1439 • The test process often remains subjective and requires an expert approach.

1440 Therefore, some tests for graphical objects, their animations, special effects should be repeated at the
1441 next phase.

1442 After a graphical object is created and tested, it is imported and tested within a test scene in the game
1443 engine to verify if it functions correctly in the game's context.

1444 The primary tests in this phase include:

1445 **1. Collision Testing:**

- 1446 • The model interacts correctly with other objects in the game.
- 1447 • The model has properly configured hitboxes.

- 1448 • Interactions between objects and object behavior adhere to the laws of physics (or the
1449 game's rules).

1450 **2. Resource Optimization:**

- 1451 • There are no clipping failures.
1452 • Level of Detail switching is properly configured.
1453 • Lighting, reflections, and shadows are correctly set up and function as intended.
1454 • There are no objects that create significant strain on the hardware of the gaming
1455 platform.

1456 **3. Special Effects Verification:**

- 1457 • Special effects do not overload the game scene or distract from the gameplay.
1458 • A special effect is synchronized with specific actions and events in the game scene.

1459 The final phase of video game graphics testing is comprehensive system testing, which often includes
1460 certification testing (for console games or after localization).

1461 As part of system testing, the integration of the graphics subsystem with other video game subsystems
1462 is verified, along with compliance with all artistic and technical requirements. In this phase, key non-
1463 functional testing is also conducted, including performance testing, compatibility testing, and load
1464 testing.

1465 Localization testing is a separate test type and will be described in detail later. However, for
1466 completeness, it should be mentioned that graphical content is one of the primary priorities when testing
1467 for legal, ethical, religious, and other regulatory compliance required for video games. Additionally,
1468 game platform developers impose specific graphical content requirements, and failure to comply may
1469 prevent a game from being published on a distribution platform.

1470

1471 **5 Audio Testing - 130 minutes**

1472 **Keywords**

1473 Defect, failure

1474 **Video Game Specific Keywords**

1475 Ambient, distortion, foley, lag, occlusion, reverb, sound effect, sound zone, volume

1476 **Learning Objectives for Chapter 5**

1477 **5.1 Principles and Concept of Audio Content in Video Games**

1478 GaMe-5.1.1 (K1) Recall the video game audio functions

1479 GaMe-5.1.2 (K1) Recall the video game audio types

1480 GaMe-5.1.3 (K2) Summarize video game audio techniques

1481 GaMe-5.1.4 (K2) Give examples of failures in video game audio content

1482 **5.2 Test Types in Audio Testing**

1483 GaMe-5.2.1 (K2) Summarize content-audio testing.

1484 GaMe-5.2.2 (K2) Summarize the approaches to technical testing of audio content.

1485 **5.3 Executing Video Game Audio Testing**

1486 GaMe-5.3.1 (K3) Apply audio testing

1487

1488 **5.1 Principles and Concepts of Audio Content in Video Games**

1489 **5.1.1 Features of Video Game Audio**

1490 Video game audio is a crucial component that influences immersion, perception, and overall player
1491 experience. In modern video games, sound design serves as a storytelling, emotional, and interactive
1492 tool.

1493 The primary objectives of game audio include supporting gameplay mechanics, creating atmosphere
1494 and emotional depth, providing information, and ensuring player feedback. It is important to recognize
1495 that sound design is not just for entertainment but also serves several essential functions:

- 1496 1. Signals. Certain sounds inform the player about resource shortages, enemy base captures,
1497 successful shots, and other important events.
1498 Spatial orientation. Sounds help players determine whether they are in a closed space or an
1499 open environment.
- 1500 2. Environmental awareness. Players navigate the game world using sounds associated with
1501 different surfaces, such as water, sand, or snow.
- 1502 3. Notifications. Sounds indicate events that are not visually displayed on the screen, such as
1503 leveling up or receiving a new quest.
- 1504 4. Atmospheric immersion. Ambient sounds enhance the feeling of presence, such as city noise,
1505 farm sounds, or battlefield chaos.

- 1506 5. Effect amplification. Sounds emphasize critical game events. For example, after a grenade
1507 explosion, the player may hear a ringing or muffled sounds, simulating a concussion effect.
1508 6. Emotions and mood. Music and sound effects evoke emotions and set the mood. For instance,
1509 combat scenes feature intense and dynamic music, while exploration is accompanied by calmer
1510 melodies.

1511 The same sounds can serve multiple functions. For example, alert or notification sounds can also
1512 contribute to the game's atmosphere.

1513 5.1.2 Video Game Audio Types

1514 Game audio can be broadly categorized into six types:

- 1515 • Background music
- 1516 • Graphical user interface sounds
- 1517 • Ambient sound
- 1518 • Character sound
- 1519 • Sound effects
- 1520 • Character voiceover

1521 **Background Music**

1522 Music is a fundamental part of the game and acts as its signature component. It sets and conveys the
1523 desired mood and can directly influence player actions—either motivating dynamic play or slowing down
1524 the pace.

1525 **Graphical User Interface Sounds**

1526 UI sounds play a functional role, providing player feedback. They confirm that an action or command
1527 has been successfully registered and processed by the game system (subsystem).

1528 **Ambient Sound**

1529 Ambient sounds reflect the characteristics of a specific location. These sounds usually play in the
1530 background and are independent of character actions. Examples include rustling leaves in a forest,
1531 ocean waves, city noise, or conversations of passersby. Such sounds immerse the player in the game
1532 world's atmosphere.

1533 **Character Sound**

1534 Character sounds (Foley) include all noises made by characters except for speech. This may include
1535 footsteps, creaking doors, jingling keys, and other effects that enhance realism.

1536 **Sound Effects**

1537 Sound effects (SFX) serve various purposes, from adding realism to creating unique audio cues, such
1538 as the sounds of magical spells or alien technology.

1539 **Character Voiceover**

1540 Voiceover is used for both playable and non-playable characters, adding variety and depth to the
1541 soundscape. It can be modified with additional effects, such as voice distortion or simulated poor
1542 reception. However, the absence of voice acting does not always negatively impact the game's
1543 atmosphere.

1544 5.1.3 Video Game Audio Techniques

1545 To create realistic and immersive audio, it is necessary to consider the spatial distribution of sound and
1546 the human auditory system's perception capabilities. For this, the following techniques are used:

1547 **Sound Zones**

1548 Sound zones are segments of the game environment where the set of sounds, their intensity, volume,
1549 and characteristics differ from the surrounding areas. Thus, sound zones help players form a clear
1550 sense of space and its characteristics.

1551 **Binaural Sound**

1552 The binaural sound effect is based on the fact that humans hear with both ears simultaneously. When
1553 a person turns their head, sound reaches one ear before the other, allowing them to determine the
1554 direction of the sound source and even estimate its distance.

1555 **Spatial Propagation**

1556 In the real world, sounds never travel unchanged—they are reflected, absorbed, or distorted depending
1557 on obstacles. If a game fails to adjust sound properties based on whether the source is behind a wall
1558 or in an open field, it breaks immersion and disrupts the atmosphere.

1559 To create a realistic soundscape, games must account for various sound propagation properties and
1560 their effects, such as occlusion, reverberation, and more.

1561 5.1.4 Types of Audio Failures

1562 Despite the wide variety of audio components in video games, failures characteristic of this subsystem
1563 can be grouped into the following categories:

- 1564 • Missing sound or sound effect,
- 1565 • Playback of incorrect sound,
- 1566 • Incorrect playback of the correct sound,
- 1567 • Sound control failures (inability to adjust character voice volume, music level).

1568 **Missing sound or sound effect**

1569 The sound does not play even though it should for a specific character action or at a particular
1570 moment.

1571 **Playback of incorrect sound**

1572 Such failures may include situations where:

- 1573 • playback of a sound that does not match the game situation;
- 1574 • playback of a sound that does not correspond to its real (historical) prototype.

1575 **Incorrect playback of the correct sound**

1576 These failures are the most numerous in their variety. They include:

- 1577 • Interruptions and cutoffs: sudden disappearance of sound or abrupt volume jumps.
- 1578 • Looping: a short sound gets stuck in infinite repetition.
- 1579 • Distortion: sounds play in a distorted and unintelligible manner.
- 1580 • Desynchronization: the sound lags behind or leads the displayed animation, object, or game
1581 situation.
- 1582 • Incorrect volume: the playback volume is either excessively high or too low.

- 1583
- 1584
- 1585
- 1586
- 1587
- 1588
- Alteration of sound properties: additional effects such as binaural sound effect, reverberation, occlusion, echo are missing.
 - Sound zone configuration: sounds from one zone are heard in another.
 - Noise, crackling: sound is overlapped by extraneous noise, crackling, hissing, clicking.
 - Noticeable “seams” between background compositions: an overly abrupt transition between background compositions.

1589 It is important to understand that some of the situations described above may be intentionally created
1590 according to the video game’s storyline (e.g., simulating poor communication quality, impaired character
1591 hearing after a concussion). Therefore, when identifying a defect, it is crucial to consider the context in
1592 which a particular sound is used.

1593 **Sound control failures**

1594 Audio files may be of perfect quality and located in the correct directories, but the player may be
1595 unable to:

- 1596
- 1597
- 1598
- 1599
- increase or decrease character voice volume,
 - adjust the background music level,
 - mute the narrator’s voice, despite such functionality being declared in the video game and available in the corresponding section of the user interface.

1600 Such failures may also be identified during UI testing.

1601 **5.2 Test Types in Video Game Audio Testing**

1602 The previous sections essentially described two groups of audio failures:

- 1603
- 1604
- 1605
1. Failures related to the placement of sound files, their linkage to in-game events, volume control, and toggling options.
 2. Failures related to the quality of audio files.

1606 This paragraph describes the following approaches to audio testing the following approaches to audio
1607 testing:

- 1608
- 1609
- Content-Audio Testing
 - Audio Technical Testing

1610 **5.2.1 Video Game Content-Audio Testing**

1611 Sound in games is a holistic system of components: character voices, musical compositions,
1612 environmental effects, UI sounds, footsteps, wind noise, engine sounds, distant battle echoes, or
1613 rustling leaves. All these components must be well-balanced and structured. If the soundscape is
1614 chaotic and unorganized, the player may feel discomfort, overload, and confusion. On the other hand,
1615 if key sounds (e.g., enemy footsteps, item pickups, important indicators) are too quiet or unclear, the
1616 player may miss crucial information, negatively impacting gameplay success.

1617 As part of content-audio testing, tests are conducted to identify defects in audio quality, as mentioned
1618 earlier. These tests focus not only on the presence of audio cues but also on their clarity, directionality,
1619 volume balance, and timing. The way sounds interact with visual cues and gameplay logic is also closely
1620 evaluated. For instance, if a warning sound doesn’t coincide with an incoming threat on-screen, it can
1621 break immersion or cause frustration.

1622 Moreover, testers examine how audio assets respond to dynamic in-game changes, such as weather
1623 transitions, day-night cycles, or player actions. Environmental reverb, occlusion, and other effects are
1624 analyzed for spatial realism.

1625 Additionally, localization testing evaluates whether translated voiceovers sync properly and preserve
1626 emotional tone. Glitches like looping defects, crackling, or missing audio are flagged during
1627 playthroughs. Overall, effective content-audio testing evaluates if the auditory environment supports the
1628 game's narrative, gameplay clarity, and emotional depth — turning a functional world into a believable,
1629 reactive one.

1630 5.2.2 Video Game Audio Technical Testing

1631 Video game audio technical testing refers to the systematic process of evaluating and verifying different
1632 qualities of audio components within a game. As video games evolve into more immersive experiences,
1633 the importance of rigorous audio technical testing cannot be overstated. Audio technical testing goes
1634 beyond simple playback checks. It covers:

- 1635 • Functional suitability: Verifying that audio triggers correctly and plays as intended during
1636 gameplay. This includes testing scripted and emergent events to evaluate if audio is
1637 contextually appropriate and responsive.
- 1638 • Consistency: Evaluating volume levels, transitions, and effects are coherent throughout
1639 different environments and hardware setups. Testing must account for audio layering, fade-ins
1640 and fade-outs, and crossfading between music tracks or ambient loops.
- 1641 • Performance: Measuring resource usage to prevent lag or audio glitches, especially in high-
1642 demand scenarios. Audio load balancing and priority systems must be evaluated under stress
1643 conditions.
- 1644 • Platform Compatibility: Testing across various devices (PCs, consoles, mobile) to verify reliable
1645 audio performance. Different drivers, APIs, and sound hardware configurations can significantly
1646 impact output and must be accounted for.

1647 These test types are also used to detect failures related to the presence of necessary audio files in the
1648 system, their correct usage in different situations, and the player's ability to control game audio within
1649 the advertised functionality (turning sounds on and off, adjusting volume levels, selecting audio
1650 languages, or switching between surround and stereo modes). Testers must also evaluate accessibility
1651 settings, verify lip-sync with dialogue, and evaluate how audio interacts with gameplay mechanics like
1652 stealth, cutscenes, or cinematic sequences.

1653 5.3 Executing Video Game Audio Testing

1654 The test types mentioned above are applied together during test activities, just as with other video
1655 game subsystems.

1656 The tester must conduct all necessary test types to obtain a complete understanding of how correctly
1657 the audio is used and configured. The number of tests depends on which objects need to be tested.

1658 Audio testing often begins at later phase of development. This is because the core game content
1659 (object/character models, maps, items) is typically created first, while the audio components are
1660 added later. By the time testing begins, the object itself usually has a completed model with
1661 animation, integrated into the client, but may not yet have sound.

1662 Audio samples should first be pre-tested by the specialists who created them (e.g., audio engineers,
1663 Foley artists, and sound effect designers). These specialists may participate in testing individual
1664 sounds before submitting them for further integration into the client.

1665 In the content-audio testing phase, which is the most extensive phase of audio testing, the tester
1666 verifies the following:

- 1667 • Presence of sound for specific game situations.
- 1668 • Relevance of sounds to the object and game scenario.
- 1669 • Absence of extraneous noise or distortion in game event sounds.
- 1670 • Avoidance of irritating sounds.
- 1671 • Appropriate audio volume levels.
- 1672 • Correct positioning of sound based on its source.
- 1673 • Proper sound zone configuration.
- 1674 • Character voice acting matches the game context and audience expectations.
- 1675 • Sounds correspond to real-world analogs (historical accuracy).
- 1676 • Overall sound balance, including musical accompaniment.
- 1677 • Failures such as looping, distortion, missing sounds, delays, or cut-offs.
- 1678 • Lack of effects that ensure a realistic audio environment.

1679 The tester must perform all of the tests listed above to develop a comprehensive assessment of the
1680 correctness of audio configuration and usage. The number of tests conducted depends on which
1681 specific objects require verification.

1682 In addition to content-audio testing, the specialist must verify the following:

- 1683 • The game allows the enabling and disabling of sounds and/or music.
- 1684 • Volume levels can be adjusted.
- 1685 • All audio files have consistent naming conventions and are stored in the correct directories
1686 within the project.
- 1687 • Audio sources and zones are correctly placed on the game map.
- 1688 • Sounds play correctly on various audio systems (e.g., headphones, speakers, studio
1689 monitors, surround systems, home theaters, and other audio formats and technologies).

1690 The final phase may include performance testing, which evaluates how audio affects the hardware
1691 resources of the gaming platform.

1692 When assessing audio environment quality, the tester must consider the following criteria:

- 1693 • The audio does not cause FPS drops in a sound-intensive scene.
- 1694 • The audio does not lead to memory leaks.
- 1695 • The set limits on the number of simultaneous sounds (e.g., max concurrent sound effects) are
1696 adhered to.

1697 If an immersive audio environment significantly impacts performance, a lightweight audio option is
1698 available for less powerful devices.

1699

1700 **6 AI and Physics Testing - 170 minutes**

1701 **Keywords**

1702 Debugging, defect, failure, load testing, performance testing, stress testing

1703 **Video Game Specific Keywords**

1704 Non-player character (NPC), pathfinding, Rigid Body Physics (RBP), role-playing game (RPG),
1705 sandbox, Soft Body Physics (SBP), video game physics

1706

1707 **Learning Objectives for Chapter 6**

1708 **6.1 Principle and Concepts of Artificial Intelligence (AI) and Physics in Video Games**

1709 GaMe-6.1.1 (K1) Recall the main aspects of video game AI

1710 GaMe-6.1.2 (K2) Give examples of failures of video game AI

1711 GaMe-6.1.3 (K2) Distinguish the main aspects of video game physics

1712 GaMe-6.1.4 (K2) Give examples of failures of video game physics

1713 **6.2 Executing AI and Game Physics Testing**

1714 GaMe-6.2.1 (K3) Apply AI testing

1715 GaMe-6.2.2 (K3) Apply video game physics testing

1716

1717

1718 **6.1 Principles and Concepts of Artificial Intelligence (AI) and Physics in** 1719 **Video Games**

1720 **6.1.1 Video Game Artificial Intelligence Features**

1721 Artificial Intelligence (AI) in video games refers to computational systems that create autonomous,
1722 responsive, and adaptive behaviors in game entities and game systems. While AI is most commonly
1723 associated with non-player character (NPC) behavior—such as enemies, allies, and neutral
1724 characters — it also encompasses broader game systems including:

- 1725 • Dynamic difficulty adjustment
- 1726 • Procedural content generation
- 1727 • Intelligent camera systems
- 1728 • Traffic and crowd simulation
- 1729 • Strategic decision-making

1730 In multiplayer-focused games, AI-controlled characters that substitute for human players are
1731 specifically referred to as "bots".

1732 The core AI algorithm follows a sequence of steps: "sense → think → act." Developers can add
1733 complexity to each phase, making NPCs appear more intelligent.

1734 To better understand the root causes of AI-related failures, it is important to know which architectures
1735 are used to implement AI in a particular game. Common AI architectures include:

- 1736 • Finite State Machine (FSM) – State-based behavior with discrete transitions
- 1737 • Behavior Tree (BT) – Hierarchical, modular decision structures
- 1738 • Utility AI – Score-based action selection using evaluation functions
- 1739 • Goal-Oriented Action Planning (GOAP) – Dynamic planning to achieve goals
- 1740 • Blackboard Systems – Shared knowledge repositories for multi-agent coordination
- 1741 • Hierarchical architectures – Layered systems combining multiple approaches.

1742
1743 Using these methods, hundreds of great games have been developed, where virtual opponents
1744 closely resemble real players. Understanding these methods helps identify weaknesses and detect
1745 defects in different situations.

1746 However, game AI is always based on predefined parameters set by developers, making its behavior
1747 predetermined and limited by the creator's design. This is a key drawback of AI architectures.
1748 Additionally, creating believable and intelligent NPC behavior requires significant development time to
1749 account for numerous scenarios and potential reactions. Due to this complexity, defects frequently
1750 occur in AI behavior, which cause AI failures.

1751 Furthermore, NPCs are usually restricted in their movement paths within game levels. To define
1752 where an NPC can navigate, developers use navigation meshes (NavMeshes). This is a data
1753 structure that describes the game world's surfaces, enabling NPCs to calculate paths from one point
1754 to another (Pathfinding).

1755 6.1.2 Types of AI Failures in Video Games

1756 AI failures in video games can significantly impact gameplay quality, immersion, and player satisfaction.
1757 These failures generally fall into the following categories:

- 1758 • Character interaction failures
- 1759 • Movement and pathfinding failures
- 1760 • Perception and awareness failures
- 1761 • Decision-making and behavioral logic failures
- 1762 • Group and swarm behavior failures
- 1763 • Performance-related failures
- 1764 • Realism and game design balance failures

1765 **Character Interaction Failures**

- 1766 • Failures in how AI-controlled characters interact with the player, environment, or other NPCs.
- 1767 • NPCs ignore player commands or presence.
- 1768 • Allies fail to support or react appropriately.
- 1769 • Characters do not use environmental components such as cover.
- 1770 • Characters behave as if unaware of environmental barriers (e.g., "see" through walls).

1771 **Movement and Pathfinding Failures**

- 1772 • Failures related to navigation and motion logic.
- 1773 • Characters become stuck in place or walk into obstacles.
- 1774 • Pathfinding defects cause navigation through inaccessible areas.
- 1775 • Characters loop endlessly in action sequences (AI stuck in loop).
- 1776 • Incorrect routes are selected due to defective navigation meshes (NavMesh).

1777 **Perception and Awareness Failures**

- 1778 • Breakdowns in sensory simulation and situational responsiveness.
- 1779 • Enemies fail to detect or respond to the player.
- 1780 • Characters forget about the player too quickly in stealth mode.
- 1781 • NPCs lack awareness of events in their surroundings.

1782 **Decision-Making and Behavioral Logic Failures**

- 1783 • Failures in action prioritization or strategic responses.
- 1784 • AI prioritizes incorrect or illogical actions.
- 1785 • Characters behave irrationally or inconsistently in similar situations.
- 1786 • Enemies attack the wrong targets (e.g., allies instead of the player).

1787 **Group and Swarm Behavior Failures**

- 1788 • Breakdowns in coordinated multi-agent behavior.
- 1789 • Crowd movements appear chaotic or unnatural.
- 1790 • AI characters in groups fail to interact with one another logically.

1791 **Performance-Related Failures**

- 1792 • AI behavior causes system performance degradation.
- 1793 • Frame rate (FPS) drops due to complex or unoptimized AI logic.
- 1794 • AI freezes or stalls in high-load or unexpected situations.

1795 **Realism and Game Design Balance Failures**

- 1796 • Confusion between defects and intentional design choices.
- 1797 • AI behaves too predictably or unrealistically.
- 1798 • Characters violate expected physical laws (e.g., float, pass through objects).
- 1799 • Overly intelligent AI frustrates players, while overly passive AI may reduce challenge.

1800 However, it is important to consider "game design conventions." Absolute realism in character behavior
1801 is not always required. For instance, an AI that is "too smart" may frustrate players.

1802 **6.1.3 Video Game Physics Features**

1803 The adherence to the laws of physics is unquestionable for most players in almost any game. Game
1804 physics serves multiple purposes, but its primary goal is to make the game intuitive and engaging. When
1805 objects behave unpredictably, players struggle to understand the rules of the game and may lose
1806 immersion or experience gameplay frustration.

1807 At the same time, as with AI, it is important to remember that the video game world has its own rules,
1808 and absolute realism can sometimes worsen the gameplay experience if all physical laws are strictly
1809 followed. Overly accurate physics simulations may hinder game flow, reduce responsiveness, or even
1810 make gameplay boring or repetitive, especially in arcade or action-heavy genres. It is crucial to
1811 remember that the main purpose of a game is entertainment, not teaching physics fundamentals.
1812 Therefore, when discussing physics failures in video games, only clear violations of key physical laws
1813 should be considered.

1814 If a game is set in a non-fantasy world, such failures are usually related to the absence of fundamental
1815 forces acting on an in-game object, such as:

- 1816 • Gravity (defect: an object levitates or shoots upward to an extreme height);
- 1817 • Inertia (defect: a fast-moving heavy object stops instantly upon abrupt deceleration);

- 1818
- Friction (defect: a heavy object slides for too long on a non-slippery surface);
- 1819
- Centrifugal force (defect: a car takes a high-speed turn on the outer lane and successfully
- 1820
- completes it without braking);
- 1821
- or cases where objects exhibit properties impossible in the real world, such as extreme speeds,
- 1822
- unrealistic elasticity, or unbreakable surfaces under massive impacts.

1823 In video games, physics is generally divided into two main categories:

1824 **Rigid Body Physics (RBP) and Soft Body Physics (SBP).**

1825 Rigid Body Physics refers to the simulation of physical behavior for objects that are assumed not to
1826 deform when forces are applied. In a game context, rigid bodies maintain a fixed shape and size while
1827 reacting to gravity, collisions, forces, impulses, and constraints according to predefined physical rules.

1828 Soft Body Physics describes forces that affect an object's shape, such as the movement of flags, water,
1829 jelly-like materials, or character clothing — essentially, any object that deforms under external forces.
1830 These effects significantly enhance realism and immersion, especially in slow-motion or cinematic
1831 sequences.

1832 The main challenge in implementing soft body physics is the massive number of calculations required.
1833 Achieving realistic effects demands millions of operations per second, often exceeding processor
1834 capabilities. As a result, soft body physics is frequently simplified. For non-essential objects, looped
1835 animations or procedural effects are often used instead of real-time physics calculations. Such
1836 simplifications are not considered failures as long as the animation appears natural and is consistent
1837 with the game's visual style and expectations.

1838 6.1.4 Types of Game Physics Failures

1839 When discussing game physics, it primarily refers to rigid body physics.

1840 All failures related to violations of physical laws in games can be broadly classified into two groups by
1841 defects that caused them:

- 1842
- Destruction failures
- 1843
- Object behavior failures

1844 Failures caused by destruction defects include:

- 1845
- Destruction without physical impact. Objects break apart without any visible reason, failing to
- 1846
- respond to player actions or environmental forces.
- 1847
- Unrealistic object destruction. Objects behave unnaturally when breaking apart.
- 1848
- Repetitive destruction. All destruction sequences occur identically, regardless of conditions.
- 1849
- Incorrect collision with destroyed objects. After destruction, players or other characters cannot
- 1850
- pass through debris.
- 1851
- Delayed or missing destruction. Destruction happens with a noticeable delay or does not
- 1852
- occur at all.
- 1853
- Incorrect disappearance of destroyed objects. After destruction, objects either instantly vanish
- 1854
- or disappear after a short delay without leaving debris.
- 1855
- Illogical destruction sequence. Object parts break apart in an illogical order or without
- 1856
- consideration of physical impact.
- 1857
- Independence from force impact. Regardless of the strength of the impact (e.g., a pistol shot
- 1858
- or a grenade explosion), destruction always looks the same.

1859

1860 Failures caused by object behavior defects include:

- 1861 • Unrealistic flight or falling physics. Objects move, fall, or bounce at unnatural speeds or
1862 trajectories.
- 1863 • Lack of gravity response. Objects do not obey gravity, remaining suspended in the air after
1864 being destroyed or moved.
- 1865 • Incorrect response to collisions and explosions. Objects do not break apart in collisions and
1866 explosions or do so without considering the force and direction of the blast wave.
- 1867 • Interaction of objects without physical contact ("Telekinesis"). Objects interact with each other
1868 without actual collision.
- 1869 • Lack of response to physical forces. Objects remain immobile even after a significant force is
1870 applied.
- 1871 • Unnatural sliding or rolling of objects. Objects that should remain stationary or behave
1872 predictably start sliding across surfaces for no apparent reason.
- 1873 • Lack of damage after collisions. When objects collide, especially at high speeds, no visible
1874 damage or deformation occurs.
- 1875 • Another failure arises when the speed of a projectile (bullet, shell, arrow) is so high that the
1876 collision with another object (especially a small one) fails to register within the system.

1877 Physics in video games is one of the most complex aspects of development. The challenge for
1878 developers is to find a balance between hardware performance limitations and realism. At the same
1879 time, it is essential to remember that a game should be engaging for players above all else. This
1880 influences the design approach to game physics — in some cases, realism must be sacrificed in favor
1881 of interesting gameplay mechanics and an improved gaming experience.

1882 6.2 Executing AI and Game Physics Testing

1883 6.2.1 Executing AI Testing in Video Games

1884 When planning AI testing, it is necessary to understand what exactly needs to be verified. The test
1885 objectives may include the verification of:

- 1886 • Functional correctness: AI must perform its tasks correctly (e.g., enemies should react to
1887 player actions, and NPCs should interact with the environment, objects, or other characters in
1888 a meaningful and consistent way).
- 1889 • Behavioral diversity: Characters and objects should exhibit natural and varied behavior,
1890 avoiding predictability. This includes non-repetitive pathing, decision-making influenced by
1891 changing context, and reactions that adapt to the player's choices.
- 1892 • Balance: AI should not be too difficult or too easy, ensuring engaging gameplay across
1893 various skill levels. AI behavior must align with player progression and be appropriate for
1894 different game modes or difficulty settings.
- 1895 • Performance: AI should operate efficiently without significantly increasing system load. Poorly
1896 optimized AI can lead to dropped frames, lag, or unresponsive game logic.

1897 At the beginning of testing, it is essential to verify basic functions that AI should perform or control. For
1898 example:

- 1899 • Interaction with the environment: AI should correctly navigate obstacles, use cover, and
1900 interact with objects.
- 1901 • Logical sequences: AI should correctly execute a sequence of actions (e.g., gathering
1902 resources, building shelters, patrolling).
- 1903 • Reaction to triggers: For example, an enemy should attack the player upon spotting them or
1904 hide when health is low.

- 1905 This activity is often automated using test scripts or built-in tools in game engines to reduce the time
1906 spent on repetitive tests.
- 1907 AI must exhibit realistic behavior. To determine this, the following aspects are tested:
- 1908 • Variety of actions in similar situations.
1909 • Randomness and variability in decision-making to prevent complete predictability.
1910 • Natural interactions with the player and the environment.
- 1911 AI should be well-balanced to keep the game engaging. For example:
- 1912 • Enemies should be intelligent enough to pose a challenge but not so difficult that defeating
1913 them becomes impossible. They should also exhibit weaknesses the player can learn and
1914 exploit.
1915 • Non-player characters (NPCs) should provide useful opportunities for the player without being
1916 overly "perfect" or monotonous, enhancing immersion through subtle, believable actions and
1917 dialog.
- 1918 During this activity, difficulty parameters are adjusted, and the overall gameplay experience is
1919 analyzed through playtesting, with real player feedback guiding further AI tuning.
- 1920 AI can significantly impact CPU and memory performance, especially in games with a large number of
1921 NPCs. Performance testing is conducted using stress tests, where:
- 1922 • A large number of AI actors are launched simultaneously, performing diverse tasks. The impact
1923 of AI processing on FPS, frame time, and resource usage is measured, ensuring game stability
1924 under high-load conditions.
- 1925 **6.2.2 Executing Game Physics Testing**
- 1926 The physics engine serves as the foundation for object interactions in a video game. Its performance
1927 directly affects the realism and credibility of the gameplay. Testing game physics is a complex
1928 process that requires detailed analysis and verification of numerous scenarios.
- 1929 It is necessary to determine which aspects of physics should be tested. The key areas include:
- 1930 • Correct object behavior: Ensuring that the engine properly implements physical laws (e.g.,
1931 gravity, friction, inertia).
1932 • Realism: Verifying that object movements and interactions look natural and do not create
1933 dissonance for players.
1934 • Performance: Evaluating the impact of the physics engine on the game's performance.
- 1935 If the game's narrative does not specify an alternate physical reality, then basic functional tests may
1936 include:
- 1937 • Gravity:
1938 ○ Objects are attracted to surfaces.
1939 ○ Objects fall with natural acceleration.
1940 • Collisions:
1941 ○ Objects correctly interact with each other and with surfaces.
1942 ○ Objects do not pass through each other.
1943 • Friction and sliding:
1944 ○ Objects slide on different surfaces depending on their mass and material
1945 composition.

- 1946 ○ Objects slow down correctly on various surfaces.
- 1947 • Other physical laws and properties:
- 1948 ○ Objects sink in water depending on their material and shape.
- 1949 ○ Objects experience air and water resistance.
- 1950 ○ Objects leave marks on different surfaces.
- 1951 The realism of the game depends on whether various game objects obey the laws of physics. To
- 1952 evaluate if that players can fully immerse themselves in realistic interactions, the following should be
- 1953 tested:
- 1954 • Realistic destruction of objects.
- 1955 • Realistic interaction between objects.
- 1956 Testing is conducted to identify the failures described above.
- 1957 The physics engine can place a significant load on the processor, especially in games with a large
- 1958 number of objects or detailed simulations. Performance is tested using:
- 1959 • Frame processing time measurements: Verifying how much time is required to calculate
- 1960 physics in each frame.
- 1961 • Memory usage measurements: Verifying how much memory is needed for physics
- 1962 calculations per frame or scene.
- 1963 These tests are performed using built-in game engine tools or specialized software.
- 1964 Video game physics testing is a meticulous process aimed at achieving realism, performance, and
- 1965 stability. Only thorough testing evaluates if physics meets player expectations and becomes an
- 1966 integral part of an immersive gaming experience.

1967 **7 Localization Testing - 150 minutes**

1968 **Keywords**

1969 Compliance, defect, failure, internationalization, localization, root cause

1970 **Video Game Specific Keywords**

1971 Cultural adaptation, historical accuracy

1972

1973 **Learning Objectives for Chapter 7**

1974 **7.1 Principles and Concepts of Video Game Localization**

1975 GaMe-7.1.1 (K1) Recall the localization core concept

1976 GaMe-7.1.2 (K2) Features of video game localization

1977 GaMe-7.1.3 (K2) Distinguish between full and partial localization

1978 GaMe-7.1.4 (K1) Recall root causes of localization defects

1979 GaMe-7.1.5 (K2) Give examples of failures of video game localization

1980 **7.2 Approaches to Localization Testing**

1981 GaMe-7.2.1 (K2) Summarize linguistic testing and adaptation testing

1982 GaMe-7.2.2 (K2) Summarize technical testing of localization

1983 **7.3 Executing Localization Testing**

1984 GaMe-7.3.1 (K3) Apply localization testing.

1985

1986 **7.1 Principles and Concepts of Video Game Localization**

1987 **7.1.1 Video Game Localization Essentials**

1988 It is essential to note that localization is not just a simple translation into multiple languages. The video
1989 game localization process includes:

- 1990 • Translating in-game dialogue texts, subtitles, tooltips, descriptions, messages, character
- 1991 names, and item names while considering regional specifics.
- 1992 • Translating cutscenes, interface components, and menus.
- 1993 • Redrawing textures and graphics.
- 1994 • Casting voice actors, dubbing, and recording audio files.
- 1995 • Integrating localized content into the game.
- 1996 • Translating and adapting the game's website and printed materials.
- 1997 • Translating marketing materials.
- 1998 • Providing post-launch support for the localized version.

1999 To mitigate risks of adapting a game for a particular region, developers use a process known as
2000 internationalization. This is the preliminary adaptation of a product for potential use by international
2001 user groups, whereas localization refers to specific modifications for a particular region or user group.

2002 Internationalization includes:

- 2003 • Designing and developing software in a way that does not create obstacles for localization.
- 2004 • Extracting localized components from the code or content so that localized versions can be
2005 loaded later or selected based on user preferences
- 2006 • Allowing the use of components that cannot be implemented before the localization process.
- 2007 • Supporting regional, linguistic, or cultural references (including fonts and formatting).

2008 This list does not necessarily include the localization of content, programs, or products. Instead, it
2009 outlines software development approaches that make future localization easier and more efficient.

2010 7.1.2 Features of Video Game Localization

2011 The difference between localization testing for video games and application software is based on the
2012 understanding that these two types of products differ significantly in several key aspects, as discussed
2013 in previous chapters.

2014 Key Differences:

- 2015 • Adaptation of graphical content
- 2016 • Adaptation of audio content
- 2017 • Adaptation of textual content

2018 As a result, these video game subsystems require mandatory adaptation, considering factors such as
2019 historical accuracy, religious, ethical, cultural, political, and ideological aspects of the target audience.

2020 Localization testers must verify that symbols, gestures, references, and dialogues are appropriate for
2021 the local context. In some cases, failure to meet these requirements may lead to ambiguous reception
2022 by the target audience or even result in the product being banned in certain countries or regions. For
2023 example, content depicting religious figures, politically sensitive flags, or controversial slogans may
2024 need to be censored or altered to comply with local regulations.

2025 Existing game development traditions, connections to literature, cinema, and other media, as well as
2026 the deep genre convergence of gaming products, require adherence to specific norms and standards.
2027 This includes preserving lore consistency, narrative tone, and thematic coherence in translated
2028 content. Violating these standards may cause negative reception among players from different
2029 regions and disrupt immersion.

2030 For example, in massively multiplayer role-playing games (MMORPGs), localization must consider
2031 established conventions for player race names, item names, and other genre-specific terminology.
2032 Consistency with prior installments or related franchises is also essential, especially for fan-favorite
2033 series. Moreover, UI layout, font size, text wrapping, and suchlike are crucial components that require
2034 thorough testing to avoid mismatches and technical defects in the localized build.

2035 Localization testers must verify that symbols, gestures, references, and dialogues are appropriate for
2036 the local context. In some cases, failure to meet these requirements may lead to ambiguous reception
2037 by the target audience or even result in the product being banned in certain countries or regions. For
2038 example, content depicting religious figures, politically sensitive flags, or controversial slogans may
2039 need to be censored or altered to comply with local regulations.

2040 Existing game development traditions, connections to literature, cinema, and other media, as well as
2041 the deep genre convergence of gaming products, require adherence to specific norms and standards.

2042 This includes preserving lore consistency, narrative tone, and thematic coherence in translated
2043 content. Violating these standards may cause negative reception among players from different
2044 regions and disrupt immersion.

2045 For example, in massively multiplayer role-playing games (MMORPGs), localization must consider
2046 established conventions for player race names, item names, and other genre-specific terminology.
2047 Consistency with prior installments or related franchises is also essential, especially for fan-favorite
2048 series. Moreover, UI layout, font size, text wrapping, and suchlike are crucial components that require
2049 thorough testing to avoid mismatches and technical defects in the localized build.

2050 7.1.3 Full and Partial Localization

2051 The scope of localization testing depends on the type of localization used by the game developers.
2052 The completeness of localization is influenced by various factors, but this syllabus does not cover
2053 them in detail. The key takeaway is that the extent of testing directly depends on the level of
2054 localization previously performed for the video game.

2055 The following types of localization can be identified:

- 2056 • Full localization / cultural adaptation
- 2057 • "Box" localization
- 2058 • Text localization
- 2059 • Localization with voiceover
- 2060 • Graphical localization

2061 **Full Localization / Cultural Adaptation**

2062 Cultural adaptation occurs when the original game is completely reworked during the localization
2063 process. Only the source code and mechanics remain unchanged, while textures, storyline, dialogues,
2064 and character models may be entirely redesigned, effectively creating a new version of the game
2065 using the same engine.

2066 This method is rarely used, but it is necessary in cases where a game cannot be released and sold in
2067 a specific market without such adaptation.

2068 **"Box" Localization**

2069 If a game is released and sold on a physical medium, the packaging text must be localized.

2070 If the game is sold digitally on an online platform, only the store page is translated, including the game
2071 description and screenshots. "Box" localization does not extend beyond this scope.

2072 **Text Localization**

2073 In most cases, all in-game text must be translated, including subtitles. This means that a player may
2074 hear dialogue in a foreign language but will see subtitles in their native language to understand the
2075 conversation.

2076 **Localization with Voiceover**

2077 In addition to text translation, character speech and dialogues are dubbed by voice actors in the target
2078 language.

2079 **Graphical Localization**

2080 Every video game contains graphical components such as textures and visual assets that may include
2081 embedded text—for example, writing on a wall, signs, newspapers, or shop banners.

2082 Graphical localization verifies that all such text within the game is translated. This may include:

- 2083 • Newspapers
- 2084 • Storefront signs
- 2085 • Road signs

2086 Since many games are set in specific locations, the localization approach may vary.

- 2087 • If an in-game object (e.g., a newspaper article) is crucial to the story, it must be translated to
- 2088 preserve key narrative components.
- 2089 • However, other text components, such as storefront signs, may remain untranslated if they do
- 2090 not significantly impact the player's experience or storyline progression.

2091 7.1.4 Root Causes of Localization Defects

2092 Even major companies are not immune to "cultural" defects in their products. In some cases, a single
2093 phrase or joke can lead to a complete ban of the game in certain countries. Such errors not only
2094 damage the game's reputation but can also lead to financial losses and public backlash.

2095 The main root causes in the localization process that can result in localization defects include:

- 2096 • Insufficient qualifications of translators, especially those without experience in the gaming
- 2097 industry or lacking creative adaptation skills.
- 2098 • Lack of knowledge about the history, culture, ethics, worldview, religion, and legislation of the
- 2099 region where the game is intended to be released, leading to misinterpretations or offensive
- 2100 content.
- 2101 • Failure to understand the connections between in-game components, storylines, episodes,
- 2102 and references to pre-existing works in literature, cinema, music, or other video games, which
- 2103 can break immersion or cause inconsistencies.
- 2104 • Technical failures (lack of support for internationalization), such as incorrect text encoding,
- 2105 font rendering failures, or broken text display in right-to-left languages.
- 2106 • Insufficient skills of the voice actors, especially in conveying tone, emotion, or character
- 2107 identity, which may distort the intended player experience and lead to player dissatisfaction or
- 2108 criticism in reviews.

2109 7.1.5 Video Game Localization Failures

2110 Considering the root causes of localization defects mentioned above, the following categories of
2111 failures can be identified:

- 2112 • Failures caused by translation defects
- 2113 • Failures related to content adaptation for regional requirements
- 2114 • Convergence failures
- 2115 • Failures related to technical aspects (including lack of internationalization support)
- 2116 • Failures related to dubbing (voiceover)

2117 Failures Caused by Translation Defects

- 2118 • Incorrect grammar, spelling, and punctuation
- 2119 • Incorrect number formatting, currency units, calendars, and dates
- 2120 • Incorrect addresses and phone numbers
- 2121 • Incorrect usage or conversion of measurement units or currencies
- 2122 • Incorrect display of numerical information
- 2123 • Incorrect proper names

- 2124 • Mistranslation of historical events, holidays, cultural references, proverbs, idioms
- 2125 geographical names (toponyms), slang, and abbreviations
- 2126 • Distortion or omission of special characters (e.g., incorrect rendering of diacritical marks in
- 2127 French, Hebrew, or Arabic)
- 2128 • Translation that does not match the context of the game
- 2129 • Translation that does not comply with platform manufacturer requirements
- 2130 • Over-localization. Not everything should be localized—some components should retain their
- 2131 original form, such as trademarks, logos, abbreviations, and product names

2132 **Failures Related to Content Adaptation for Regional Requirements**

- 2133 • Failures caused by ignoring different cultural perceptions of humor, satire, and jokes,
- 2134 including the acceptability of humor regarding specific subjects.
- 2135 • Failures due to misunderstanding the target audience's views on religious topics, including
- 2136 religious objects, rituals, and ceremonies, as well as attitudes toward marginal cults (e.g.,
- 2137 Satanism, paganism).
- 2138 • Historical inaccuracy or failure to consider alternative interpretations of historical events by
- 2139 the target audience. This includes:
 - 2140 ○ One-sided portrayals of historical events, military conflicts, and scientific discoveries.
 - 2141 ○ Misrepresentation of existing objects, cultural heritage, or traditional lifestyles of
 - 2142 specific nations.
- 2143 • Failures caused by a misinterpretation of national culture and worldview. This includes
- 2144 ignoring:
 - 2145 ○ National stereotypes
 - 2146 ○ Traditional cuisine
 - 2147 ○ Symbols and color meanings
 - 2148 ○ Clothing styles
 - 2149 ○ Lifestyles of different ethnic groups
 - 2150 ○ Attitudes toward children, LGBTQ+ communities, and animal cruelty
- 2151 • Failures related to legal restrictions, including compliance with:
 - 2152 ○ Age ratings
 - 2153 ○ Child protection laws
 - 2154 ○ Religious regulations
 - 2155 ○ Restrictions on violence, drugs, sexual content, race, terrorism, political criticism, and
 - 2156 government opposition
 - 2157 ○ Bans on specific symbols and imagery

2158 **Convergence Failures**

- 2159 • Loss of recognizability and identity of a literary, cinematic prototype, or other cultural
- 2160 phenomenon after localization.
- 2161 • Inconsistencies in established terminology and naming conventions between the video game
- 2162 and its literary, cinematic, or other original references.
- 2163 • Legal failures related to license violations and copyright infringement.

2164 **Failures Related to Technical Aspects (Including Lack of Internationalization Support)**

- 2165 • Corrupted Text / Garbled Characters. Corrupted text appears when decoding text with an
- 2166 incorrect character encoding, resulting in random symbol substitutions, often from a different
- 2167 writing system. This usually makes the text unreadable.
- 2168 • Localized text exceeding interface constraints (cut-off or scrolling failures). A term translated
- 2169 from one language to another may have a different character length, leading to truncated text
- 2170 in UI components.

- 2171 • Misalignment. After localization, UI layouts may require reconfiguration to preserve the
2172 original alignment.
- 2173 • Missing line-by-line translation. Testers should evaluate dialogue box text, images, and
2174 screenshots in documents or the user interface to verify all content is properly localized
2175 according to user expectations.
- 2176 • Overlapping components. Occurs when UI components overlap, making text or controls
2177 unreadable or inaccessible.
- 2178 • Missing text. Some text may be lost during translation or development.
- 2179 • Incorrect font/size. Different countries use different default fonts and sizes, which may not
2180 affect functionality but can significantly impact user experience by making text difficult to read.
- 2181 • Incorrect hotkeys. In some cases, a hotkey may be unavailable because the corresponding
2182 letter does not exist in the localized language or keyboard layout.
- 2183 • Variables in text templates. Different language types (analytical, synthetic) use various
2184 declensions, cases, and plural forms that affect word structure. A common error among
2185 developers is splitting sentences containing variables, leading to incorrect grammar in the
2186 target language. Developers often fail to account for how variables change sentence structure
2187 in certain languages.
- 2188 • Audio sequence desynchronization between original and translated content. Translated
2189 character phrases may be longer or shorter than the original due to differences in syllable
2190 count. Since cutscenes in games are often pre-recorded, localization teams must carefully
2191 balance translation accuracy with syllable count while ensuring lip-sync correctness with
2192 character animations.

2193 **Failures Related to Dubbing (Voiceover)**

- 2194 • Unnatural-sounding speech.
- 2195 • Inappropriate voice casting for characters.
- 2196 • Failure to account for national and individual speech characteristics of characters.

2197 One of the most obvious consequences of incorrect localization is the deterioration of the user
2198 experience. Poor translation, grammatical errors, or cultural mismatches can lead to
2199 misunderstanding the storyline, character traits, and game mechanics. Players may struggle with
2200 completing quests or navigating the interface, reducing their enjoyment and increasing frustration
2201 levels.

2202 Incorrect cultural adaptation can result in misinterpretations or even offense to the target audience.
2203 This not only harms the perception of the game but may also lead to boycotts, lawsuits, or other legal
2204 consequences.

2205 **7.2 Test Types in Localization Testing**

2206 Localization testing involves verification of the content of a video game application for compliance with
2207 linguistic and cultural requirements, as well as the specifics of a particular country or region. These
2208 test types help identify localization defects or translation errors in the localized version before the final
2209 product reaches the user.

2210 To optimize the localization test process, different test types are used, depending on the roles
2211 involved and the level of parallel execution. Specifically:

- 2212 1. Linguistic Testing (translation, spelling, grammar) and Adaptation Testing (cultural, religious,
2213 legal adaptation).
- 2214 2. Technical Testing (internationalization and other technical aspects).

2215 7.2.1 Linguistic Testing and Adaptation Testing

2216 Linguistic testing involves a series of tests to verify that:

- 2217 • Consistent terminology is used throughout the game.
- 2218 • There are no grammatical errors.
- 2219 • There are no spelling errors.
- 2220 • Punctuation rules are correctly followed.
- 2221 • The correct text direction is applied (right-to-left or left-to-right).
- 2222 • The translation is accurate and considers the risks and causes of localization failures
- 2223 described earlier.
- 2224 • Correct names for trademarks, cities, locations, job titles are used.

2225 At the same time, the tester's task includes evaluating the quality of video game content to meet the
2226 target region or user group requirements. Testing must carefully consider all previously mentioned
2227 risks and factors.

2228 Critical areas for content appropriateness and compliance:

- 2229 • Humor
- 2230 • Religion, including occultism
- 2231 • History
- 2232 • Morality and ethics
- 2233 • Prejudices and stereotypes related to culture and people
- 2234 • Politics
- 2235 • Regional cultural norms
- 2236 • Superstitions and symbolism (including color and number symbolism)
- 2237 • Attitudes toward sex and LGBTQ+ representation
- 2238 • Attitudes toward children and the elderly
- 2239 • Legal restrictions
- 2240 • Connections between the video game and literary, cinematic, or other media sources

2241 Given the strict control over political correctness in the modern world, it is essential to take this aspect
2242 seriously during localization testing.

2243 Within this test type, video and audio content must also be tested to verify:

- 2244 • Whether the audio matches the game component it belongs to.
- 2245 • Whether the audio corresponds to the character's gender.
- 2246 • Sound clarity (absence of interference, proper synchronization of tone between the original
2247 and translated versions, and balanced volume levels).
- 2248 • Authenticity of the audio, including historical accuracy.
- 2249 • Stylistic and cultural aspects of the audio (accents, speech characteristics).

2250 7.2.2 Localization Technical Testing

2251 The objective of technical testing is to verify internationalization and technical aspects of localization.

2252 Technical testing has the to evaluate all game components after the integration of localized content to
2253 verify that they function correctly. This test type considers the aspects mentioned earlier, including
2254 text display, interface functionality, audiovisual synchronization, subtitle correctness, audio file
2255 playback, and overall performance. If these aspects are overlooked, players may encounter failures
2256 that degrade their gaming experience, such as missing subtitles, broken UI layouts, incorrect audio
2257 triggering, or system crashes tied to specific language settings.

2258 Technical aspects of localization are particularly scrutinized during certification testing for console
2259 games. Manufacturers of these gaming platforms impose strict and region-specific requirements on
2260 the technical aspects of localization. These include conformity with button naming conventions, error
2261 messaging formats, UI alignment, save/load prompts, and behavior under network disconnection.
2262 Failure to comply with these requirements may result in the game being banned from publication on a
2263 specific platform or delayed until all defects are resolved.

2264 Before a game is released on consoles, it must undergo mandatory certification tests, which
2265 necessarily include:

- 2266 • Evaluation of correctness in texts, the interface, or voiceovers, verifying that there are no
2267 display overflows, truncated strings, or incorrect speaker attributions.
- 2268 • Verification of the proper adaptation of system notifications and the game's user interface,
2269 including menu flow, confirmation dialogs, and region-appropriate terminology.
- 2270 • Verification of the functional correctness of the localized version according to the console's
2271 regional settings, including correct time/date formats, number systems, and language
2272 switching features across all supported regions.

2273 7.3 Executing Localization Testing

2274 In the process of localization testing, the following activities can be identified. Some of these may be
2275 conducted simultaneously:

- 2276 • Analysis and planning of localization testing
- 2277 • Execution of technical testing
- 2278 • Linguistic testing and adaptation test execution

2279 Analysis and Planning of Localization Testing

2280 This includes:

- 2281 • Providing testers with all necessary documentation related to the test product.
- 2282 • Analyzing the original version of the video game and any previously localized versions (if
2283 available).
- 2284 • Creating a glossary and translation memory to help testers correctly interpret the terminology
2285 used.
- 2286 • Compiling a database of game-related components and references to verify accurate
2287 translation and adaptation.
- 2288 • Collecting materials describing all key characteristics of the target audience to assist in
2289 evaluating the adequacy of localization adaptation.
- 2290 • Selecting and configuring defect management tools — a document or platform where all
2291 defects identified during localization testing will be recorded.

2292 Execution of Technical Localization Testing

2293 The following activities are performed:

- 2294 • Verification of technical capabilities for internationalization.
- 2295 • User interface testing.
- 2296 • Subtitle testing (including synchronization with audio).
- 2297 • Testing of regional language support (including text direction and special characters).
- 2298 • Testing for missing texts and translations.
- 2299 • Testing of audio and video content (including verifying that the audio corresponds to the
2300 relevant game components, ensuring the character's voice matches their gender).

- 2301
- Additional technical tests.

2302 **Linguistic Testing and Adaptation Test Execution**

2303 This is the most complex and critical test type. It involves various tests carried out by a wide range of
2304 specialists. For example:

- 2305
- Testing the functional correctness of the translation for all game content (including
2306 compliance with platform manufacturers' requirements, translation language rules, translation
2307 traditions).
 - Testing the consistency of terminology.
 - 2308
 - Verifying the correct usage of measurement units (including currency).
 - 2309
 - Testing for stylistic and genre appropriateness (including references to source materials).
 - 2310
 - Reviewing the game's visual, audio, and textual content for historical accuracy and alignment
2311 with real-world prototypes.
 - 2312
 - Testing the compliance of localized content with the norms and requirements of the target
2313 region.
 - 2314

2315 In different companies, the phases of localization testing may vary, including or excluding specific
2316 tests or even entire localization test effort depending on the situation. The test process described
2317 above serves as a basic framework and can be adapted to fit a specific project.

2318 Additional tests may be conducted, such as:

- 2319
- User interface testing
 - 2320
 - Graphics testing
 - 2321
 - Audio testing
 - 2322
 - Certification testing

2323 **8 Game Controllers Testing - 55 minutes**

2324 **Keywords**

2325 Compliance, defect, ergonomics testing, failure, interoperability, security

2326 **Video Game Specific Keywords**

2327 Gamepad, stick, video game controller

2328 **Learning Objectives for Chapter 8**

2329 **8.1 Principles and Concepts of Game Controllers**

2330 GaMe-8.1.1 (K2) Distinguish input devices and their functions.

2331 GaMe-8.1.2 (K2) Give examples of failures of video game controllers.

2332 **8.2 Approaches to Game Controller Testing**

2333 GaMe-8.2.1 (K1) Recall controller interoperability testing.

2334 GaMe-8.2.2 (K1) Recall controller performance testing

2335 GaMe-8.2.3 (K1) Recall controller ergonomics testing

2336 GaMe-8.2.4 (K1) Recall controller compliance testing

2337 GaMe-8.2.5 (K1) Recall controller usability testing

2338

2339 **8.1 Principles and Concepts of Video Game Controllers**

2340 **8.1.1 Types of Video Game Controllers**

2341 A video game controller is an input device used in video games to control the movement and actions of
2342 game objects, typically a character (or one of the characters) in the game. Video game controllers
2343 provide more intuitive and precise control compared to standard keyboards and mice.

2344 Below are the main types of game controllers:

Video Game Controller	Functionality
Gaming Keyboards and Mice	Specialized keyboards and mice with additional programmable buttons, high precision, and adjustable weight.
Gamepads (Joysticks)	The most common type of one-handed or two-handed controllers, featuring multiple buttons and analog sticks for movement and actions.

Steering Wheels, Pedals, and Gear Shifters	Primarily used in racing games, providing a more realistic driving experience.
VR Controllers	Designed for use with virtual reality systems, allowing hand movement tracking and interactive capabilities in a virtual environment.
Specialized Controllers	Various devices such as dance pads, music controllers, and sports simulation devices designed for specific game genres.
Motion Controllers	Controllers equipped with accelerometers for motion tracking.
Touchscreens	An input and display interface that allows users to interact directly with on-screen content by touching the screen surface.

2345

2346 The purpose of video game controllers is to provide smooth and intuitive control over the gameplay,
2347 enhance the user experience, and increase player enjoyment.

2348 The main functions of a controller include:

- 2349 • Ensuring interaction between the player and the game
- 2350 • Enhancing convenience and ergonomics
- 2351 • Increasing immersion and realism
- 2352 • Supporting a variety of game genres
- 2353 • Ensuring accessibility and inclusivity

2354 Controller testing focuses on verifying how video game software interprets and responds to input from
2355 supported devices. From a game testing perspective, this includes validating input handling,
2356 responsiveness, configuration, and in-game behavior across supported platforms and control schemes.
2357 Testing of controller hardware, firmware, or driver implementation itself is typically outside the scope of
2358 game testing. When input-related failures are observed, testers focus on determining whether the cause
2359 lies within the game software's handling of input or in external components, and report findings
2360 accordingly.

2361 8.1.2 Failures Related to the Performance of Video Game Controllers

2362 It is important to note that this syllabus does not cover physical failures of game controllers caused by
2363 manufacturing defects or improper user handling.

2364 Common failures include:

- 2365 • Input lag
- 2366 • Driver incompatibility
- 2367 • Limited customization
- 2368 • Incorrect button mapping
- 2369 • Failures with additional features (vibration, accelerometer functionality)

2370 **Input Lag**

2371 This refers to the delay between the moment a player performs an action (e.g., pressing a button or
2372 moving a stick) and when that action is displayed on the game screen. This delay is measured in
2373 milliseconds (ms) and directly affects control responsiveness. Such a failure is particularly noticeable
2374 in video games that require fast player reactions.

2375 **Driver Incompatibility**

2376 From a game testing perspective, driver incompatibility refers to situations where the video game
2377 software is unable to correctly interpret or respond to input provided through the controller's software
2378 interface. This may manifest as failure to recognize the device, incorrect operation of control elements,
2379 or input lag during gameplay.

2380 While such failures may be influenced by external factors such as drivers, firmware, or operating system
2381 behavior, testing in this context focuses on identifying whether the observed failure originates from the
2382 game software's handling of controller input or lies outside the scope of the game application. Findings
2383 are reported accordingly, rather than attempting to validate or correct hardware, firmware, or driver
2384 implementations.

2385 **Limited Customization**

2386 This refers to situations where video game controller customization options are restricted due to
2387 software defects, driver failures preventing players from adjusting settings to their individual
2388 preferences.

2389 **Incorrect Button Mapping**

2390 Commands entered by the player are misinterpreted by the system. For example, pressing one button
2391 may trigger an action assigned to a different button, or the command may not register at all. This failure
2392 can be related to defective drivers, outdated controller firmware.

2393 **Failures with Additional Features (Vibration, Accelerometer Functionality)**

2394 The causes of such failures vary, including driver incompatibility, conflicts with other hardware or
2395 software, or firmware defects in the controller.

2396 Typical failures of video game controllers are usually caused by outdated device drivers, conflicts with
2397 software from other connected devices, or firmware defects within the controller itself. These factors
2398 determine the approach to testing controllers.

2399 **8.2 Test Types in Game Controller Testing**

2400 Video game controller testing typically begins when the game functionality is ready for use with a
2401 standard game controller.

2402 Game controller testing may include:

- 2403 • Connecting/disconnecting the game controller to/from a PC or console.
- 2404 • Using the game controller with a low battery level.
- 2405 • Using one or multiple game controllers simultaneously.
- 2406 • Unconventional use of the game controller (negative testing).
- 2407 • Testing vibration functionality (presence and intensity level).

2408 8.2.1 Interoperability Testing

2409 If an application requires the use of a game controller as an input device, testing must verify that all
2410 software functions, interface interactions, and gameplay mechanics must correctly interact with the
2411 connected device. The user must be able to control both the user interface components and the
2412 gameplay itself without switching controllers or experiencing lag in input recognition.

2413 The assigned control components—buttons, analog sticks, steering wheel rotations, voice commands,
2414 or spatial movements—must correspond to the intended actions of the game character. This includes
2415 menu navigation, camera control, movement, combat actions, and inventory management. The tester
2416 must also verify that when the video game controller is disconnected from the device, the application
2417 pauses automatically if possible or displays a contextual prompt to inform the player.

2418 If the game is implemented in real-time and cannot be paused, disconnecting the video game controller
2419 should not result in the player being disconnected from the server, dropped from a match, or losing
2420 progress. Instead, it must be accompanied by an informational message, giving the player an
2421 opportunity to reconnect the device or switch to an alternative control scheme without disrupting the
2422 session. Resilience, usability, and user experience continuity are critical criteria in this test phase.

2423 8.2.2 Performance Testing

2424 Performance testing of video game controllers focuses on measuring the responsiveness and reliability
2425 of input signals under various conditions. Key aspects include input latency—the time between a
2426 player's action (e.g., button press or joystick movement) and the corresponding on-screen response—
2427 as well as consistency in signal transmission over time. This testing is especially critical in fast-paced
2428 genres like fighting games, first-person shooters, or racing simulations, where even slight delays can
2429 affect gameplay accuracy, reaction timing, or competitive balance.

2430 Testers assess the video game controller's behavior under different system loads, network conditions
2431 (for wireless video game controllers), and platform configurations to identify performance bottlenecks
2432 or irregularities. Key performance indicators include frame delay, packet loss, signal interference, and
2433 input jitter. Additionally, testers must evaluate how quickly the system detects repeated inputs, such as
2434 rapid button presses (e.g., in quick-time events or combo sequences), and whether analog inputs like
2435 pressure sensitivity, dead zones, and tilt responsiveness remain stable over time.

2436 Long-duration sessions are also tested to look for overheating, battery drain, input drift, or disconnect
2437 frequency, especially in Bluetooth or proprietary wireless video game controllers. Comprehensive
2438 performance testing evaluates if players experience fluid, predictable, and competitive gameplay, free
2439 of hardware-induced delays or inconsistencies.

2440 8.2.3 Ergonomics Testing

2441 The buttons on video game controllers from the most popular brands have established functions. The
2442 X, A, and , X buttons on dual-handed video game controllers are very often used in applications as
2443 "confirm," "accept selection," or "interact" with an interactive component, while the B and buttons
2444 function as "cancel," "decline," or "return." These conventions form a critical part of user interface
2445 consistency and player muscle memory, especially in genres like action-adventure, RPGs, and
2446 platformers.

2447 Button combinations are configured and managed by software logic within the game engine. When
2448 testing button combinations, the tester must pay attention to whether they are anatomically comfortable
2449 for different groups of players (depending on genre, age, dominant hand, hand size, disability): whether
2450 the player can press multiple buttons simultaneously or sequentially without discomfort or misinputs.

2451 Additionally, testers must assess the timing sensitivity of combos, the customizability of control
2452 schemes, and whether key remapping options are provided for accessibility. Games should also
2453 accommodate controller latency, ensuring that multi-button actions register correctly across different
2454 hardware types, such as wired versus wireless controllers. Proper validation of button use and layout
2455 contributes to player comfort, accessibility, and inclusive design.

2456 8.2.4 Testing Video Game Controllers for Compliance

2457
2458 The tester must validate that the developer uses traditional layouts for popular video game controllers,
2459 ensuring user familiarity and input consistency across genres. For example, as mentioned earlier, the
2460 X, A, and , X buttons are used for "accept," while the B and buttons function as "cancel." This
2461 convention is especially important for menu navigation, dialog confirmations, and interaction prompts.

2462 On a keyboard, the W, A, S, and D keys are used for movement, Ctrl or C for crouching or crawling,
2463 and the spacebar for jumping. These mappings are standard in most PC games, especially first-person
2464 shooters, RPGs, and platformers.

2465 The left mouse button is traditionally assigned for shooting, attacking, or selecting units in strategy
2466 games, while the right mouse button is typically reserved for aiming, camera control, or movement
2467 commands. The tester must verify that any deviation from these norms is clearly documented,
2468 customizable, and intuitively introduced, especially for new players.

2469 Incorrect mapping or lack of input flexibility can result in frustration, mistimed actions, and accessibility
2470 defects, particularly for players relying on muscle memory or assistive control schemes. Thorough
2471 validation verifies that control schemes meet user expectations and support a wide range of playstyles
2472 and skill levels.

2473 8.2.5 Usability Testing

2474 Usability testing for video game controllers focuses on evaluating how effectively and comfortably
2475 players can interact with the game using the controller. This includes assessing the intuitiveness of
2476 button layouts, ease of access to essential controls, and the overall ergonomics of gameplay
2477 interactions. Testers verify whether players can easily navigate menus, customize input settings, and
2478 perform in-game actions without confusion or discomfort. Special attention is given to accessibility
2479 features, such as remappable controls, support for alternative input devices, and usability by players
2480 with different physical abilities. The goal is to evaluate that the video controller provides a responsive,
2481 inclusive, and frustration-free experience across diverse player profiles and gaming scenarios.

2482 Testers must also verify that the use of any video game controller does not give players a significant
2483 advantage over others. Since dual-handed video game controllers are significantly inferior to the
2484 keyboard/mouse combination in terms of aiming speed and precision, they often include an aim assist
2485 feature that helps target enemies and track them automatically. The reticle (aiming mark or crosshair)
2486 simply "sticks" to the enemy. In this case, the tester's responsibility is to monitor how far the reticle
2487 follows the enemy and whether it locks onto their head, which is often a vulnerable area where a hit
2488 deals additional damage.

2489

2490

9 References

2491

9.1 Standards

2492 [ISO25000] ISO/IEC 25000:2014, Systems and software engineering — Systems and software
2493 Quality Requirements and Evaluation (SQuaRE) — Guide to SQuaRE

2494 [ISO25010] ISO/IEC 25010:2011, Systems and software engineering — Systems and software
2495 Quality Requirements and Evaluation (SQuaRE) — System and software quality models

2496

9.2 ISTQB documents

2497 [ISTQB_AL_SEC] ISTQB Advanced Level Security Testing Syllabus, Version 2016

2498 [ISTQB_ALTA_SYL] ISTQB Advanced Level Test Analyst Syllabus, Version 3.1.2

2499 [ISTQB_ALTTA_SYL] ISTQB Advanced Level Technical Test Analyst Syllabus, Version 4.0

2500 [ISTQB_ALTM_SYL] ISTQB Advanced Level Test Manager Syllabus, Version 2012

2501 [ISTQB_CTFL_MAT] ISTQB Mobile Application Tester, Version 2019

2502 [ISTQB_EXAM_S&R] ISTQB Exam Structure and Rules, Game Testing, Version 1.0

2503 [ISTQB_FL_AT] ISTQB Foundation Level Agile Tester Syllabus, Version 2014

2504 [ISTQB_FL_PT] ISTQB Foundation Level Performance Testing Syllabus, Version 2018

2505 [ISTQB_FL_SYL] ISTQB Foundation Level (Core) Syllabus, Version 2018

2506 [ISTQB_GLOSSARY] ISTQB Glossary of Terms used in Software Testing,
2507 <https://glossary.istqb.org/>

2508 [ISTQB_UT_SYL] ISTQB Foundation Level Usability Testing Syllabus, Version 2018

2509

9.3 Books

2510 [Nystrom14] Nystrom, R. (2014). Game programming patterns. Genever Benning., ISBN: 978-
2511 0990582908. URL: <https://www.gameprogrammingpatterns.com/>

2512 [Gregory18] Gregory, J. (2018). Game engine architecture. AK Peters/CRC Press., ISBN: 978-
2513 1466560017. URL: <https://www.gameenginebook.com/>

2514 [Buttfield19] Buttfield-Addison, P., Manning, J., & Nugent, T. (2019). Unity game development
2515 cookbook: essentials for every game. O'Reilly Media., ISBN: 9781491999158

2516 [Lee16] Lee, J. (2016). Learning unreal engine game development. Packt Publishing Ltd., ISBN:
2517 9781784398156

2518 [Tavakkoli18] Tavakkoli, A. (2018). Game Development and Simulation with Unreal Technology. CRC
2519 Press., ISBN-13: 978-1498706247

2520 [Romero19] Romero, M., & Sewell, B. (2019). Blueprints Visual Scripting for Unreal Engine: The faster
2521 way to build games using UE4 Blueprints. Packt Publishing Ltd., ISBN: 9781789347067

2522 [Chandler11] Chandler, H. (2011). The Game Localization Handbook 2nd Edition, Jones & Bartlett
2523 Learning, ISBN: 0763795933

2524 [Retsker81] Retsker, Ya. I. (1981). Textbook for translation from English into Russian. M.:
2525 Prosveschenie. (In Russian).

2526 9.4 Links (Web/Internet)

2527 Note: All references are current as of April 28, 2022.

- 2528 • [ISTQB-Web] <https://www.istqb.org/>
- 2529 • [URL1]
2530 [https://research.ncl.ac.uk/game/mastersdegree/workshops/technicalrequirementschecklists/T](https://research.ncl.ac.uk/game/mastersdegree/workshops/technicalrequirementschecklists/Technical%20Requirements%20Checklist%20Workshop.pdf)
2531 [echnical%20Requirements%20Checklist%20Workshop.pdf](https://research.ncl.ac.uk/game/mastersdegree/workshops/technicalrequirementschecklists/Technical%20Requirements%20Checklist%20Workshop.pdf)
- 2532 • [URL2] [https://docs.microsoft.com/en-us/gaming/gdk/_content/gc/live/get-started/live-xbl-](https://docs.microsoft.com/en-us/gaming/gdk/_content/gc/live/get-started/live-xbl-overview)
2533 [overview](https://docs.microsoft.com/en-us/gaming/gdk/_content/gc/live/get-started/live-xbl-overview)
- 2534 • [URL3] <https://gamepad-tester.com/>
- 2535 • [URL4] [https://igda-website.s3.us-east-2.amazonaws.com/wp-](https://igda-website.s3.us-east-2.amazonaws.com/wp-content/uploads/2021/04/09142137/Best-Practices-for-Game-Localization-v22.pdf)
2536 [content/uploads/2021/04/09142137/Best-Practices-for-Game-Localization-v22.pdf](https://igda-website.s3.us-east-2.amazonaws.com/wp-content/uploads/2021/04/09142137/Best-Practices-for-Game-Localization-v22.pdf)

2537

10 Appendix A – Learning Objectives/Cognitive Level of Knowledge

2538
2539

2540 The following learning objectives are defined as applying to this syllabus. Each topic in the syllabus
2541 will be examined according to the learning objective for it.

2542 The learning objectives begin with an action verb corresponding to its cognitive level of knowledge as
2543 listed below.

2544 Level 1: Remember (K1)

2545 The candidate will remember, recognize and recall a term or concept.

2546 **Action verbs:** Recall, recognize.

Examples
Recall the concepts of the test pyramid. 2548
Recognize the typical objectives of testing.

2549
2550

Level 2: Understand (K2)

2551 The candidate can select the reasons or explanations for statements related to the topic, and can
2552 summarize, compare, classify and give examples for the testing concept.

2553 **Action verbs:** Classify, compare, differentiate, distinguish, explain, give examples, interpret,
2554 summarize

Examples	Notes
Classify test tools according to their purpose and the test activities they support.	
Compare the different test levels.	Can be used to look for similarities, differences or both.
Differentiate testing from debugging.	Looks for differences between concepts.
Distinguish between project and product risks.	Allows two (or more) concepts to be separately classified.
Explain the impact of context on the test process.	
Give examples of why testing is necessary.	
Infer the root cause of defects from a given profile of failures.	
Summarize the activities of the work product review process.	

2555
2556

Level 3: Apply (K3)

2557 The candidate can carry out a procedure when confronted with a familiar task, or select the correct
2558 procedure and apply it to a given context.

2559 **Action verbs:** Apply, implement, prepare, use

Examples	Notes
Apply boundary value analysis to derive test cases from given requirements.	Should refer to a procedure / technique / process.
Implement metrics collection methods to support technical and management requirements.	
Prepare installability tests for mobile apps.	
Use traceability to monitor test progress for completeness and consistency with the test objectives, test strategy, and test plan.	Could be used in a LO that wants the candidate to be able to use a technique or procedure. Similar to 'apply'.

2560

2561
2562
2563
2564
2565

11 Appendix B – Business Outcomes Traceability Matrix with Learning Objectives

This section lists the traceability between Certified Tester Game Testing Business Outcomes and Certified Tester Game Testing Learning Objectives.

Business Outcomes: Certified Tester Game Testing	GaMe-BO1	GaMe-BO2	GaMe-BO3	GaMe-BO4	GaMe-BO5	GaMe-BO6
GaMe-BO1 Describe basic concepts of video game testing	16					
GaMe-BO2 Determine video game risks, goals and requirements under the needs and expectations of stakeholders		13				
GaMe-BO3 Conceptually design, implement and execute basic video game tests			15			
GaMe-BO4 Know the approaches to video game testing and their purpose				9		
GaMe-BO5 Recognize the tools supporting video game testing					7	
GaMe-BO6 Determine how test activities align with the software development lifecycle and reduce the cost of developing and publishing video games						2

2566

			GaMe-BO1	GaMe-BO2	GaMe-BO3	GaMe-BO4	GaMe-BO5	GaMe-BO6
1. The Specifics of Video Game Testing								
GaMe-1.1.1	K1	Recognize specifics of video game testing	x					
GaMe-1.1.2	K1	Recall product risks in video game software		x				
GaMe-1.1.3	K1	Recall how video game risks can be mitigated by testing.		x				
GaMe-1.1.4	K2	Give examples of specific failures related to video game testing		x				
GaMe-1.1.5	K2	Explain the specific test objectives of video game testing	x			x		
GaMe-1.2.1	K1	Recognize specific roles and tasks in the video game development team	x			x		
GaMe-1.3.1	K1	Recall test activities throughout the video game software development lifecycle						x
2. Testing Game Mechanics								
GaMe-2.1.1	K2	Differentiate the testing of gameplay mechanics and non-gameplay mechanics						
GaMe-2.1.2	K2	Differentiate the testing of core mechanics and meta mechanics	x					
GaMe-2.1.3	K2	Differentiate the testing of client, server, and client-server mechanics	x					
GaMe-2.1.4	K2	Give examples of failures in video game mechanics		x				
GaMe-2.2.1	K3	Apply test types and test approaches of game mechanics testing			x	x	x	
3. Video Game Level Testing								

GaMe-3.1.1	K2	Distinguish the video game levels components	x					
GaMe-3.1.2	K2	Give examples of failures in video game levels.		x				
GaMe-3.2.1	K2	Summarize geometry testing.			x			
GaMe-3.2.2	K2	Summarize the playtesting.			x			
GaMe-3.3.1	K3	Apply level testing.				x		
4. Graphics Testing								
GaMe-4.1.1	K1	Recall the types of video game graphic components	x					
GaMe-4.1.2	K2	Give examples of failures in video game graphics		x				
GaMe-4.2.1	K1	Recall artistic testing			x		x	
GaMe-4.2.2	K1	Recall technical testing of graphics.			x		x	
GaMe-4.2.3	K1	Recall playtesting.			x			
GaMe-4.3.1	K3	Apply graphics testing				x		
5. Audio Testing								
GaMe-5.1.1	K1	Recall the video game audio functions	x				x	
GaMe-5.1.2	K1	Recall the video game audio types	x					
GaMe-5.1.3	K2	Summarize video game audio techniques	x					
GaMe-5.1.4	K2	Give examples of failures in video game audio content		x				
GaMe-5.2.1	K2	Summarize content-audio testing.			x			
GaMe-5.2.2	K2	Summarize the main test types to technical testing of audio content.			x			
GaMe-5.3.1	K3	Apply audio testing				x		
6. AI and Physics Testing								
GaMe-6.1.1	K1	Recall main aspects of video game AI	x					
GaMe-6.1.2	K2	Give examples of failures of video game AI		x				
GaMe-6.1.3	K2	Distinguish the main aspects of video game physics	x					
GaMe-6.1.4	K2	Give examples of failures of video game physics		x				
GaMe-6.2.1	K3	Apply video game AI testing				x	x	
GaMe-6.2.2	K3	Apply video game physics testing				x		
7. Localization Testing								
GaMe-7.1.1	K1	Recall the essentials of localization	x					
GaMe-7.1.2	K2	Distinguish the main types of video game localization	x					
GaMe-7.1.3	K2	Distinguish between full and partial localization	x					
GaMe-7.1.4	K1	Recall root causes of localization defects		x				
GaMe-7.1.5	K2	Give examples of failures of video game localization		x				
GaMe-7.2.1	K2	Summarize linguistic testing and adaptation testing			x			
GaMe-7.2.2	K2	Summarize technical testing of localization			x			
GaMe-7.3.1	K3	Apply localization testing				x	x	
8. Game Controllers Testing								
GaMe-8.1.1	K2	Distinguish typical and specialized input devices and their functions	x					

GaMe-8.1.2	K2	Give examples of failures of video game controllers.		x				
GaMe-8.2.1	K1	Recall controller functionality testing.			x			
GaMe-8.2.2	K1	Recall controller security testing			x			
GaMe-8.2.3	K1	Recall controller ergonomics testing			x			
GaMe-8.2.4	K1	Recall controller compliance testing			x			
GaMe-8.2.5	K1	Recall controller usability testing			x			

2567

12 Appendix C – Release Notes

2568

2569 ISTQB® Certified Tester Game Testing Syllabus v2.0 is a major update. For this reason, there are no
2570 detailed release notes per chapter and section. However, a summary of principal changes is provided
2571 below.

2572 This major release has made the following changes:

2573 Syllabus structure

2574 All learning objectives have been edited to make them atomic and to create one-to-one traceability
2575 from learning objectives to content to avoid having content without a corresponding learning objective.
2576 Each learning objective is described in a section with the same number (e.g., GaMe-1.1.1 is
2577 discussed in Section 1.1.1). The goal was to make this version more homogeneous, practice oriented
2578 and to be up to date in trends of industry (e.g. by including AI-related topics), easier to read,
2579 understand, learn, and translate, focusing on increasing practical usefulness and the balance between
2580 knowledge and skills. An important reason of the update is to align the content with the CTFL v4.0.1
2581 which was published after CT-GaMe v1.0.1. All chapters are now organized in the following structure:

- 2582 • Principles and concepts of game testing area,
- 2583 • Test types for this testing area,
- 2584 • Executing testing for this area.

2585 There are 51 learning objectives in v2.0 compared to 55 in v1.0.1. The table below shows detailed
2586 statistics on the number of learning objectives and the training time.

Syllabus version	#K1 LO (time)	#K2 LO (time)	#K3 LO (time)	#Total LO (time)
Current (2.0)	19 (95 min)	25 (375 min)	7 (420 min)	51 (890 min)
Previous (1.0.1)	12 (100 min)	39 (585 min)	4 (240 min)	55 (925 min)

2587

2588 Changes in learning objectives

- 2589 • The K2 GaMe-1.1.4 was transformed to “Give examples of specific failures related to video
2590 game testing” without changing K-level.
- 2591 • Two K2 GaMe-2.1.1 and GaMe-2.1.2 were merged into one K2 GaMe-2.1.1 on gameplay
2592 mechanics and non-gameplay mechanics.
- 2593 • Chapter 5 on game level testing was moved before chapter 3 on graphics testing and chapter
2594 4 on sound testing.
- 2595 • Old K2 GaMe-5.2.1 was replaced with K3 GaMe-3.3.1, old K2 GaMe-5.2.2 was removed.
- 2596 • New sections 3.2.1 and 3.2.2 related to geometry testing and playtesting were added on
2597 game levels testing chapter.
- 2598 • Chapter on sound testing was renamed to “5. Audio testing”.
- 2599 • K-level of old GaMe-3.3.1 on executing graphics testing was changed to K3 and LO was
2600 renumbered to GaMe-5.3.1.
- 2601 • Chapter 6 on AI-related topics was added with appropriate new K1-K3 LOs.
- 2602 • Old K2 GaMe-7.3.3 was split into three LOs: K2 GaMe-7.2.1 and GaMe-7.2.2, K3 GaMe-
2603 7.3.1.
- 2604 • Old chapter 6 was renumbered to 8. Game Controllers Testing.
- 2605 • Four K1 LOs were added: GaMe-8.2.1 to GaMe-8.2.4

2606

13 Appendix D – Game Testing Specific and other Terms

Term	Definition
3D model	A digital representation of an object or character built using three-dimensional geometry.
ambient	The sound of the environment that accompanies a certain location, situation, or phase of a video game.
animation	A technique for creating the illusion of moving images using a sequence of still images and replacing each other at a high frequency.
binaural sound effect	An effect that refers to the perception of three-dimensional sound created when audio is presented separately to each ear, mimicking how humans naturally hear in real-world environments.
block-out	See grey box
client-side mechanics	Video game mechanics that work on the video game client side. See also video game mechanics.
clipping	A graphical failure where a character or object passes through another object or environment component in a visually unrealistic way, ignoring physical boundaries.
collision	The interaction that occurs when two or more objects within the game world occupy the same space or come into contact with each other.
color coding	The process of marking objects with different colors as a means of identification.
computer-generated imagery (CGI)	Static and moving images generated using three-dimensional computer graphics.
concept phase	The phase in the game software lifecycle in which the core concepts and initial design of the game product are created.
core mechanics	Video game mechanics that define the intended game experience for a player.
crowdtesting	A software testing approach in which a product is tested by a large group of external testers from different countries, with varying levels of experience, devices, operating systems, and usage scenarios, rather than solely by an internal QA team.
cultural adaptation	The process of adapting video game environment conditions to create the characteristic features of a given culture.
cutscene	Non-interactive or semi-interactive scripted sequences that temporarily take control away from the player to present narrative, exposition, character development, or major story events.
finite state machine	A computational model used to represent and control behavior by defining a finite set of states, transitions between those states, and rules that determine when transitions occur.
Foley	The process of creating and recording custom sound effects that are synchronized with on-screen actions to enhance realism, clarity, and immersion in a video game.
gamepad	A type of two-handed game controller used with video game consoles. See also video game controller.
gameplay	The core interactive experience and mechanics that define how a player interacts with the game.
gameplay mechanics	Video game mechanics that players consciously interact with, directly influencing gameplay.
game character	A person or any other entity acting in a game.
game design	The process of creating the form and content of the gameplay of a video game being developed.
game designer	A person responsible for developing the rules and content of the gameplay.

Term	Definition
game engine	The software in which all other components of a video game are built.
game environment	The content of a game: gameplay, game level, game objects, and in some cases, the game story.
grey box	An early prototype of a game level that uses simple, untextured geometry to block out the basic structure, layout, and flow of the environment before any final art or detailed assets are added. See block-out,
historical accuracy	Compliance of in-game content to its corresponding content in a real life.
hitbox	An invisible shape or defined area around characters, objects, or components in the game environment used to detect collisions.
immersion	A player's deep sense of involvement, presence, and emotional engagement within the game world.
lag	A delay in video game performance due to a slow Internet connection or memory leakage or severe CPU/RAM/HDD usage.
Level balance	The process of adjusting and fine-tuning the difficulty, pacing, resource distribution, enemy placement, and overall challenge curve within a specific game level to verify a fair, engaging, and consistent gameplay experience.
level design	A phase in video game development that involves creating video game levels and missions.
level editor	The software that is used to create and edit locations and environments within a video game.
level of detail (LoD)	A technology in which a simplified copy of an object is created to display the same object as viewed at a different distance.
loot	In-game items or resources that a player receives as a result of specific actions or exploring the game world.
mapping	The process of projecting and assigning 2D texture data onto a 3D model's surface.
meta mechanics	Video game mechanics that define how a player is intended to use a video game. See also video game mechanics.
multi-platform	The ability to use software on different platforms.
narrative	A story conveyed through video game mechanics.
non-gameplay mechanics	Mechanics that players cannot influence or can only partially affect within a game.
non-player character (NPC)	A character in a video game controlled by the game engine or an artificial intelligence system.
occlusion	A phenomenon in which one object partially or completely blocks another object from the view of the camera or from a sound source, affecting the visual or auditory perception of a scene.
particles	Elementary visual objects controlled by a particle system and used in large numbers to create dynamic visual effects that are not represented as separate gameplay entities.
pathfinding	The process in which non-player characters or entities determine the optimal route from one point to another within a game world.
player versus player (PvP)	A video game mode in which a player interacts with one or more players.
post-production phase	The phase in the game software lifecycle after release in which a game product is distributed, marketed, and maintained.
pre-production phase	The phase in game software lifecycle in which a prototype is developed and the feasibility of a game product is evaluated.
production phase	The phase in the game software lifecycle in which a game product is developed.

Term	Definition
quest	A video game genre that focuses on exploring the game world, interacting with characters, and solving logical puzzles to progress through the storyline.
racing wheel	A type of two-handed video game controller used to imitate a steering wheel, pedals and gear lever.
reverb	The persistence of sound after the original sound source has stopped.
Rigid Body Physics (RBP)	A computational modeling of objects that do not deform when forces are applied.
role-playing game (RPG)	A genre of video games in which a player assumes the roles of characters in a fictional setting.
sandbox	A game design style that offers players a high degree of freedom to explore, interact, and create within a virtual world.
scene lighting	The amount, size, color, and harshness of light surrounding a character to match the scene of a video game.
server-side mechanics	Video game mechanics that work on the video game server. See also video game mechanics.
setting	The video game environment in which the action takes place.
Soft Body Physics (SBP)	A computational modeling of an object that can deform, bend, compress, stretch, or jiggle when forces are applied.
sound effect	A sound other than speech or music made artificially for a video game object.
sound zone	A bounded area within a game environment where a specific sound effect or a set of audio parameters is applied.
speedrunning	The practice of completing a video game or a specific level as quickly as possible with the goal of setting a time record.
stick	A video game controller whose mobility is limited to two degrees of freedom. See also video game controller.
store	A service that provides digital distribution of video games and other game content.
structural geometry	The technology that provides the terrain relief and the surface on which the video game characters can move.
terrain	A collection of components related to a playing surface on which a character in a video game moves.
texture	A raster image superimposed on the surface of a polygonal model to give it color or imitate relief.
trigger	One or more actions that initiate an event.
video game	An electronic game in which a player controls images on a video screen.
video game console	An electronic device designed for a video game.
video game controller	A device used to provide input to a video game.
video game level	A separate area within a virtual game world where a player must complete a specific task.
video game mechanics	The rules, systems, and interactions that define how a video game operates and how players interact with it
visual effect (VFX)	The creation or manipulation of any on-screen imagery that does not physically exist in real life.

2607

14 Appendix E – Index

2608

To be done after technical editing.