ЖИТОМИРСЬКА ПОЛІТЕХНІКА
ДЕРЖАВНИЙ УНІВЕРСИТЕТ

ТЕСТУВАННЯ ІГОР

ЛЕКЦІЯ 12

ПЛАН

1. Життєвий цикл програмного забезпечення
2. Що таке тестування?
3. Мета тестування і рівні тестування
4. Відмінності тестування ПЗ від тестування ігор
5. Категорії багів в іграх
6. Типи тестування
7. Техніки тест дизайну
8. Тестова документація

ЖИТТЄВИЙ ЦИКЛ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Життєвий цикл програмного забезпечення — сукупність окремих етапів робіт, що проводяться у заданому порядку протягом періоду часу, який починається з вирішення питання про розробку програмного забезпечення і закінчується припиненням використання програмного забезпечення

Зазвичай до етапів життєвого циклу відносять:

- Аналіз вимог
- Проектування
- Програмування
- Тестування і налагодження
- Експлуатацію, супровід і підтримку

ЩО ТАКЕ ТЕСТУВАННЯ?

Перевірка відповідності між реальною поведінкою програми та її поведінкою на кінцевому наборі тестів, обраному певним чином.

[IEEE Guide to Software Engineering Body of Knowledge, SWEBOK, 2004]

ЦІЛІ ТЕСТУВАННЯ

· Виявлення дефектів
· Підвищення впевненості в рівні якості
· Надання інформації для прийняття рішень
· Запобігання дефектів

Основні цілі тестування - оцінити відповідність системи зазначеним потребам.

РІВНІ ТЕСТУВАННЯ

Тестування на різних рівнях проводиться протягом усього життєвого циклу розробки і супроводу ПЗ. Рівень тестування визначає те, над чим проводяться тести: над окремим модулем, групою модулів або системою в цілому. Проведення тестування на всіх рівнях системи – це запорука успішної реалізації та здачі проекту.

Рівні тестування (*Testing levels*):

- Компонентне або Модульне тестування (*Component testing or Unit testing*)
- Інтеграційне тестування (*Integration testing*)
- Системне тестування (*System testing*)
- Приймальне тестування (*Acceptance testing*)

МЕТОДИ ПРИЙМАЛЬНОГО ТЕСТУВАННЯ:

Тестування замовником самостійно. Це ризиковано в тому плані що у замовника може не бути творчих ресурсів, а завантаження по поточним завданням може розтягти процес приймання.

Тестування (Аудит) третьою стороною. Наймається спеціалізована компанія на тестуванні або підписується договір з конкурентом постачальника на надання послуг аудиту. Оптимально.

Спільне тестування за сценаріями із замовником. Постачальник допомагає готувати пакет матеріалів для приймального тестування, готує команду замовника до методичного приймального тестування, контролює хід приймального тестування і терміни його виконання. Присутність інженера з тестування з боку виконавця допоможе краще зафіксувати розбіжності, зауваження та виявлені дефекти.

ЩО ТАКЕ ТЕСТУВАННЯ ІГОР?

Тестування ігор – це процес перевірки якості гри.

Тестування ігор - це складніший процес, ніж тестування неігрових додатків, оскільки ігри відрізняються наявністю найчастіше складної ігрової механіки.

З ЧОГО СКЛАДАЄТЬСЯ ГРА?

Ігрові механіки

Графіка

Анімація

Історія, нарратив

Саунддизайн

Монетизація

Аналітика

…. тощо

ІГРОВІ МЕХАНІКИ

Ігрова механіка – це набір правил, як щось працює. При виконанні A, відбувається B. Якщо A вірне, то можна зробити B. Іншими словами, механіка – це правила, які зачіпають гравців, ігрові частки, ігровий стан і взагалі всі способи зміни ігрового стану

Одні й ті ж механіки будуть актуальними для великої кількості ігор. Особливо це стосується поділу за жанрами і типами. Наприклад, всі RPG повинні володіти певними базовими механіками, без яких вони просто не зможуть належати до цього жанру. Те ж актуально і для шутерів, RTS і абсолютно всіх ігор.

1. Досягнення (Achievement)

Визначення: Віртуальне або фізичне втілення будь-якого звершення. Зазвичай їх розглядають як нагороди.

Приклад: Значок, рівень, нагорода, бали, по суті все, що можна прийняти за нагороду може бути ачівкою.

2. Механіка призначеної зустрічі (Appointment Dynamic)

Визначення: Динаміка, в якій для досягнення успіху необхідно повернення в заздалегідь певний момент часу для вжиття будь-яких дій. Механіку призначеної зустрічі часто об'єднує міцний зв'язок з режимами інтервальних винагород або виборчих.

Приклад: При поверненні в гру в певний час, користувач отримує якесь заохочення.

3. Уникнення (Avoidance)

Визначення: Акт, що спонукає гравця до дій, за допомогою не винагороди, а уникнення покарання. Викликає ряд послідовних дій узгоджених за часом з розкладом.

Приклад: Натискайте важіль кожні 30 секунд, щоб не отримати удар струмом.

4. Поведінковий контраст (Behavioral Contrast)

Визначення: Теорія, яка визначає те, як сильно може змінюватися поведінка відповідно до змін очікувань.

Приклад: За виконання квесту гравцеві дають 100 досвіду в перший раз. Надалі – по 5000. При виконанні цього ж квесту, скажімо, в десятий раз – йому дають 100 досвіду. Як наслідок – швидше за все гравець перестане виконувати цей квест. Є ще варіант з життя: Щоб зробити добре, потрібно спочатку зробити погано, а потім – як було.

Насамперед потрібно визначитися, яку саме ігрову механіку необхідно протестувати. Дуже важливо розуміти, ЩО саме перевіряти, а потім – ЯК це зробити.

Після того, як буде визначена ігрова механіка та її складові елементи, потрібно починати вибудовувати послідовний ланцюжок тестів, які максимально торкатимуться всіх аспектів цієї механіки.

Необхідно пам'ятати про додавання перевірок на взаємодію різних механік між собою. Важливо опрацювати всі допустимі варіації взаємодії майбутнього користувача з механікою, а також визначити вплив, який вона буде проводити весь ігровий процес

Основна механіка гри - колекціонування та підбір героїв для боїв проти підземелля або інших гравців на арені

В чому складність її тестування?

ПРИКЛАД

Кожен герой має кілька характеристик і набору скілів. При застосуванні практично всі скіли можуть надавати кілька ефектів: шкода, бафф, дебафф, хіл, таунт. У цей момент відбувається зміна параметрів та поточного стану гри. До того ж, ефекти взаємодіють між собою, мають шанси спрацьовування і можуть залежати від інших.

Покрокова битва, одночасно в ній можуть брати участь до п'яти героїв з кожного боку, іноді навіть більше (наприклад, бос може закликати помічників).У кожного героя по три-чотири власні скілла плюс сети артефактів, що додають одну-дві пасивні здібності, плюс таланти (теж пасивки, по 15 на героя).

В одній битві може одночасно виявитися:10 [герої] × ((3 [скіли] + 1 [артефакти] + 15 [таланти]) × 3 [ефекти в скілі]) = 10 × (19 × 3) = 570 ефектів

Візмемо 10 героїв з повтореннями із 370 отримаемо

$$\overline{C}_{370}^{10} = C_{10+370-1}^{10} = \frac{379!}{10! \cdot 369!} = 14949526844210984025$$

ПРИКЛАД

ОСОБЛИВОСТІ ТЕСТУВАННЯ МОБІЛЬНИХ ІГОР

На мобільних девайсах є певні умовності, які у малому або у великому об'ємі здатні впливати на якість кінцевого продукту.

1 особливість мобільних платформ – розмір екрану. Наприклад - *Перекриття елементів керування вирізом*

2 особливість мобільних ігор – якість мобільного інтернету.

3 особливість – продуктивність. Найважливіший, але іноді ігнорований розробниками до останніх стадій розробки, пункт. Сюди можна віднести швидкість запуску ігрових локацій, фреймрейт під час idle сцен, фреймрейт під час завантажених сцен, фреймрейт на складних локаціях.

4 особливість мобільних ігор – жести, та й сам Touch-інтерфейс у цілому.
Використовуючи відомі механіки Touch/Swipe/Multi Touch, розробники отримують можливість зробити інтуїтивне та зрозуміле керування. Але з великими можливостями приходить велика відповідальність. Кожен елемент управління – це окремий модуль, який може зламатися за допомогою найнесподіваніших способів.

Наприклад: кнопка може виявитися не клікабельною, або вічно затиснутою; анімація скрола після свайпа може відображатися ривками; після багаторазового або одночасного натискання декількох різних елементів можна отримати конфлікт функцій.

5 особливість мобільних ігор – порушення у роботі з боку системи.

Наприклад -
Системні переривання:

- вихід з програми;
- згортання додатку;
- дзвінки;
- будильники;
- увімкнена музика;
- сповіщення;
- розрядження батареї.

КЛАСИФІКАЦІЙ БАГІВ ЗА СЕРЙОЗНІСТЮ

Blocker (Блокуючі):
Баги, які повністю зупиняють прогрес гри. Наприклад, гра крашиться при запуску або неможливо завершити місію.
Приклад: Гравець застряє на екрані завантаження і не може продовжити.
Critical (Критичні):
Серйозні баги, які порушують основний геймплей, але не зупиняють гру повністю.
Приклад: Персонаж провалюється крізь текстури або зникає ключовий предмет.
Major (Суттєві):
Баги, що впливають на геймплей або візуальний досвід, але мають обхідні шляхи.
Приклад: Зброя не завдає шкоди ворогам.
Minor (Несуттєві):
Невеликі проблеми, які майже не впливають на геймплей.
Приклад: Некоректне відображення тексту або дрібні графічні артефакти.
Trivial (Дрібні):
Малозначущі дефекти, які майже не впливають на гравця.
Приклад: Помилки в тексті опису об'єктів.

КЛАСИФІКАЦІЙ БАГІВ ЗА ТИПОМ

Функціональні баги (Functional Bugs):
Помилки, пов’язані з порушенням функціональності гри.
Приклад: **Кнопка в меню не працює або місія не завершується після виконання умов.**

Графічні баги (Graphics/Visual Bugs):
Помилки, пов’язані з відображенням гри: моделі, текстури, анімації тощо.
Приклад: **Некоректне накладення текстур, "зламані" моделі персонажів, флікеринг об’єктів.**

Аудіо баги (Audio Bugs):
Проблеми зі звуком у грі.
Приклад: **Відсутність звукових ефектів або зациклення фонової музики.**

Продуктивність (Performance Bugs):
Помилки, які впливають на швидкодію гри.
Приклад: **Падіння FPS у певних локаціях або довгі затримки завантаження.**

Інтерфейсні баги (UI/UX Bugs):
Проблеми з інтерфейсом гри.
Приклад: **Некоректно відображається меню або елементи накладаються один на одного.**

Мережеві баги (Network Bugs):
Помилки, пов'язані з онлайн-частиною гри.
Приклад: **Розриви з'єднання, лаги під час гри в мультиплеєрі.**

Логічні баги (Logic Bugs):
Помилки в ігрових механіках або поведінці персонажів.
Приклад: **NPC поводиться нелогічно, наприклад, атакує стіну замість гравця.**

Фізичні баги (Physics Bugs):
Проблеми з фізикою об'єктів у грі.
Приклад: **Персонаж літає замість того, щоб ходити, або автомобіль не реагує на зіткнення.**

Баги дизайну рівнів (level design):
Невидима стіна, відсутність геометрії (текстура присутня, але колізії моделі немає, що дозволяє пройти крізь стіну);

Штучний інтелект (artificial intelligence):
Гравець не в змозі рухатися правильно по ходу гри, не рухається зовсім, занадто часто вмирає, не може відкрити двері;

Стабільність (stability):
Фрізи, краш (чорний екран), Crash to Desktop (ПК), неможливо завантажити рівень, гра не відповідає;

ЗА ПЛАТФОРМАМИ:

Платформо-залежні баги (Platform-Specific Bugs):
Помилки, які виникають лише на певних платформах.
Приклад: **Гра крашиться на консолях, але працює стабільно на ПК.**

Кросплатформенні баги (Cross-Platform Bugs):
Помилки, які з'являються на всіх платформах.
Приклад: **Відсутність звуку незалежно від пристрою.**

ЗА ВПЛИВОМ НА ГЕЙМПЛЕЙ:

Експлойти (Exploits):
Помилки, які гравці можуть використовувати для отримання несправедливих переваг.
Приклад: **Нескінченний запас ресурсів через помилку в механіці.**

Баги руйнування прогресу (Progression Bugs):
Помилки, які блокують проходження гри.
Приклад: **Неможливість завершити завдання через відсутність потрібного NPC.**

Косметичні баги (Cosmetic Bugs):
Баги, які не впливають на геймплей, але псують зовнішній вигляд гри.
Приклад: **Некоректне відображення тіней або зміщення елементів інтерфейсу.**

ПРИКЛАД

ТИПИ ТЕСТУВАННЯ

1. Функціональне тестування (Functional Testing)

Перевіряє, чи відповідає гра функціональним вимогам.

- **Перевірка основних ігрових механік (рух персонажа, бойова система, взаємодія з об’єктами).**
- **Тестування меню, налаштувань, збереження та завантаження гри.**
- **Перевірка виконання завдань і прогресу в грі.**

2. Нефункціональне тестування (Non-Functional Testing)

Оцінює аспекти гри, які не стосуються основної функціональності.

- **Тестування продуктивності (Performance Testing):**
  Перевіряє стабільність FPS, швидкість завантаження рівнів, споживання ресурсів (пам’ять, процесор).
- **Тестування стресу (Stress Testing):**
  Імітація екстремальних умов, наприклад, великий потік гравців у мультиплеєрі.
- **Тестування на збої (Crash Testing):**
  Перевіряє, як гра поводиться у випадках аварійного завершення (наприклад, при різкому вимкненні живлення).

3. Ручне тестування (Manual Testing)

Тестувальники вручну перевіряють різні аспекти гри, від геймплея до графіки.

- **Виявлення візуальних або аудіо багів.**
- **Перевірка логіки гри та її відповідності очікуванням.**
- **Тестування на відповідність дизайну.**

4. Автоматизоване тестування (Automated Testing)

Застосування спеціальних скриптів або програм для перевірки гри.

- **Перевірка коректності роботи окремих модулів (юніт-тести).**
- **Тестування великих сценаріїв, наприклад, автоматичне проходження рівнів.**
- **Повторювані задачі, які займають багато часу вручну (наприклад, перевірка відтворення тисяч анімацій).**

5. Тестування геймплея (Gameplay Testing)

Зосереджується на перевірці ігрового досвіду:

- **Баланс ігрових механік.**
- **Реіграбельність та рівень задоволення від гри.**
- **Тестування взаємодії між персонажами або гравцями в мультиплеєрі.**

6. Тестування користувацького досвіду (UX Testing)

Оцінює зручність і комфорт гравця під час гри:

- **Чи інтуїтивно зрозумілий інтерфейс?**
- **Чи легко новачкам освоїти механіки гри?**
- **Чи відповідає дизайн гри очікуванням гравців?**

7. Тестування на сумісність (Compatibility Testing)

Перевіряє, як гра працює на різних платформах і пристроях:

- **Тестування на ПК із різною конфігурацією (процесори, відеокарти, операційні системи).**
- **Перевірка роботи на консолях, мобільних пристроях.**
- **Оцінка сумісності з різними роздільними здатностями екрану.**

8. Тестування локалізації (Localization Testing)

Перевіряє, чи гра правильно адаптована до різних мов і культур.

- **Переклад текстів, їхнє відображення (наприклад, чи вміщаються тексти у вікна).**
- **Адаптація елементів гри до локальних стандартів (дата, валюта, одиниці вимірювання).**
- **Перевірка роботи мов із різним написанням (наприклад, ієрогліфи, справа наліво).**

9. Тестування безпеки (Security Testing)

Особливо актуальне для онлайн-ігор.

- **Виявлення вразливостей, які можуть призвести до зломів.**
- **Захист облікових записів гравців і особистих даних.**
- **Перевірка захисту від чітів та зловживань.**

10. Регресійне тестування (Regression Testing)

Перевірка того, що нові зміни або виправлення не зламали вже працюючі функції.

11. Альфа- та бета-тестування (Alpha and Beta Testing):

- **Альфа-тестування: Проводиться всередині команди розробників або з вузьким колом тестувальників.**
- **Бета-тестування: Відкрите або закрите тестування із залученням реальних гравців для отримання зворотного зв'язку.**

12. Експлоративне тестування (Exploratory Testing)

Тестування без чітких сценаріїв, коли тестувальник досліджує гру, намагаючись знайти несподівані баги.

13. Соціальне тестування (Social Testing):

Перевірка мультиплеєрних або соціальних функцій гри:

- **Робота чату та списків друзів.**
- **Перевірка взаємодії гравців у кооперативі чи змаганнях.**

ПЛАТФОРМИ

- Комп'ютери , ноутбуки
- VR
- Браузерні ігри
- Ігри в соціальних мережах
- Мобільні пристрої (смартфони, планшети)
- VR
- AR
- Браузерні ігри
- Ігри в соціальних мережах
- Консолі
- VR
- Портативні консолі

ТЕХНІКИ ТЕСТ ДИЗАЙНУ

Плейтестинг (Playtesting)
Основна техніка, яка передбачає, що розробники або тестувальники грають у гру, оцінюючи її на предмет зручності, задоволення та збалансованості. Типи плейтестингу: *Внутрішній тестинг, Зовнішній тестинг, Фокус-групи*

A/B тестування (A/B Testing)
Використовується для порівняння двох версій одного елемента гри (наприклад, інтерфейсу або механіки) і допомагає визначити, який з варіантів краще працює для гравців. Це особливо корисно для виявлення найбільш привабливих варіантів балансу або дизайну.

Баланс-тестинг
Техніка, яка аналізує ігровий баланс, зокрема відносну силу персонажів, зброї або тактик. Тестувальники перевіряють, щоб у грі не було «домінуючих стратегій», які роблять інші підходи малоефективними, і щоб кожен елемент мав чіткі сильні та слабкі сторони.

Тестування на передбачуваність і реактивність

Перевіряє, чи інтуїтивно зрозумілі наслідки дій гравця, чи вони передбачувані, і чи отримує гравець зворотний зв'язок від гри, який допомагає розуміти результати своїх дій.

Рандомізація і тест на вдачу

Перевіряє вплив випадкових елементів (наприклад, генерації предметів, дій супротивників) на загальний ігровий досвід, щоб випадковість не робила гру повністю непередбачуваною, але вносила потрібний рівень інтересу.

Петля зворотного зв'язку (Feedback Loop Testing)

Динамічні механіки, що допомагають відстаючим гравцям або навпаки посилюють лідера, підлягають тестуванню на те, чи створюють вони захопливіші та напруженіші моменти в грі, не роблячи результат занадто випадковим.

Тестування прогресу (Progression Testing)

Перевірка того, наскільки добре відчувається прогрес у грі. Тестувальники оцінюють, чи гравець не застряє на рівнях, чи отримує достатню винагороду за досягнення, і чи гра не стає надто складною або надто легкою з часом.

UX/UI тестування

Перевірка інтерфейсу користувача для забезпечення простоти, зручності та інтуїтивності. Це тестування також допомагає перевірити, наскільки легко новачки можуть освоїти гру та розібратися з елементами управління.

Тестування на реіграбельність

Важливо оцінити, чи хочеться гравцям повертатися до гри після завершення. Тестувальники шукають механіки, що підтримують інтерес — варіативність, багатозначність рішень, різноманітність шляхів розвитку або можливість спробувати нові стратегії.

ТЕСТОВА ДОКУМЕНТАЦІЯ

Тестова документація — це набір документів, що описують процеси, сценарії, методи та результати тестування програмного забезпечення. У геймдизайні та розробці ігор тестова документація допомагає забезпечити якість гри, її збалансованість, стабільність та відповідність вимогам.

Тест-план (Test Plan)

Головний документ, що описує загальний підхід до тестування. Включає:

- **Мета тестування.**
- **Обсяг і межі тестування (які функції або аспекти гри будуть перевірятися).**
- **Стратегії тестування (наприклад, мануальне тестування, автоматизоване тестування).**
- **Ресурси та ролі (хто відповідальний за виконання).**
- **Розклад і етапи тестування.**
- **Ризики та способи їхнього зниження.**

Тестові сценарії (Test Scenarios)

Це високорівневі описи ситуацій, які потрібно протестувати. Наприклад:

- **Тестування механіки бою.**
- **Перевірка прогресу персонажа під час розвитку.**
- **Тестування взаємодії з інтерфейсом.**

Чек-листи (Checklists)

Списки дій, які потрібно виконати для перевірки певних функцій. Вони використовуються для швидкого тестування без написання повноцінних тест-кейсів.
Приклад чек-ліста:

- **Перевірити, чи зберігається прогрес після виходу з гри.**
- **Перевірити, чи працює основне меню.**
- **Перевірити, чи відображається правильна кількість очок після завершення місії.**

Тест-кейси (Test Cases)

Детальні інструкції для виконання конкретних тестів. Тест-кейс включає:

- **Ідентифікатор тест-кейсу.**
- **Назву тесту.**
- **Попередні умови (що має бути виконано до початку тесту).**
- **Кроки виконання тесту.**
- **Очікуваний результат.**
- **Реальний результат (заповнюється під час виконання тесту).**
- **Статус тесту (пройшов/не пройшов).**

Приклад:
Назва тест-кейсу: Перевірка правильності нарахування очок за перемогу.

- **Попередні умови: Гравець має завершити рівень.**
- **Кроки: Завершити рівень із результатом 90%.**
- **Очікуваний результат: Гравець отримує 900 очок.**

Звіти про дефекти (Bug Reports)

Документи, що описують знайдені помилки або недоліки. Зазвичай включають:

- **Ідентифікатор дефекту.**
- **Опис проблеми.**
- **Кроки для відтворення.**
- **Очікуваний та реальний результат.**
- **Скріншоти або відео (за необхідності).**
- **Статус дефекту (відкрито/виправлено/відхилено).**
- **Пріоритет і серйозність.**

Звіт про тестування (Test Report)

Підсумковий документ, що містить результати тестування. Він може включати:

- **Загальний статус тестування.**
- **Кількість знайдених дефектів (і їхній статус).**
- **Процент виконаних тестів.**
- **Рекомендації щодо подальшого вдосконалення гри.**

Матриця відповідності (Traceability Matrix)

Це таблиця, яка показує зв’язок між вимогами, тест-кейсами та знайденими дефектами. Вона дозволяє переконатися, що всі вимоги були протестовані, а всі дефекти — виправлені.

ВМІТИ

Складати mind map, test-case, check-list, bug report

Використовувати різні види тестування

Тестувати всі види графіки в іграх

Тестувати та складати вимоги

Тестувати ігри будь-якого жанру на різних двигунах

Оцінювати та оптимізувати свою роботу

Проводити мінімальний набір тестування продуктивності та безпеки

Працювати з підтримкою