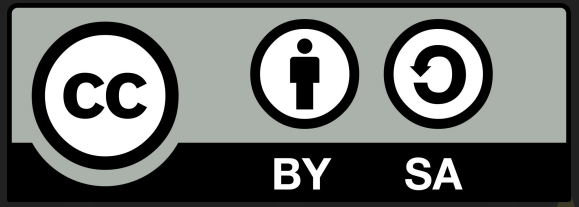
Play Test e Bug Fixes

Slides por:
Breno M. F. Viana (brenov@usp.br)



Este material é uma criação do
Time de Ensino de Desenvolvimento de Jogos Eletrônicos (TEDJE)
Filiado ao grupo de cultura e extensão Fellowship of the Game (FoG), vinculado ao ICMC - USP

Play Test

Testes em Jogos

- Jogos digitais são programas de computador;
- Testes automatizados:
  - Unitário;
  - Integração;
  - Etc.
- Contudo, isso não é suficiente para jogos.

Assassin's Creed Valhalla - FAIL Compilation!

Fonte: https://www.youtube.com/watch?v=afrikMJvsM4.

Play Test

- Teste de gameplay;
- Identificação de problemas técnicos (game testers);
- Feedback sobre experiência (público externo);
- Tipos de Play Tests:
  - Interno;
  - Externo Fechado;
  - Externo Aberto.

Play Test Interno

- Identificação (game tester):
  - Comportamento incorreto.
- Relatório (game tester):
  - Descrição do estado do jogo.
- Análise (game designer/developer):
  - Reprodução do problema e correção.
- Verificação (game tester):
  - Comportamento incorreto corrigido.

O que testar no Play Test?

- Áudio;
- Física;
- Gráficos (ex: realismo);
- Multiplayer/networking;
- Nível/mundo do jogo;
- IA;
- Balanceamento;
- API de modificações (mods);
- **Fator de diversão e experiência.**

Metodologias de Play Test

- Testes de funcionalidade;
- Testes de localização;
- Testes de compatibilidade;
- Testes de conformidade:
  - Requerimentos/performance.
- Testes de regressão:
  - Verifica se antigos bugs voltaram a aparecer.

Tipos de Testes de Funcionalidade

- Testes *ad-hoc*;
- Casos de teste;
- Teste de caixa preta;
- Teste de caixa branca.

Testes *ad-hoc*

- Testar intuitivamente coisas que um usuário pode fazer;
- Testar possíveis problemas no jogo;
- É necessário muita criatividade;
- Não é garantido testar todas as funcionalidades;
- Cada testador irá focar em sua área de preferência;
- Teste interno ou externo.

Casos de Teste

- Produzir lista de testes a serem realizados:
  - Funções do produto;
  - Quais funções interagem entre si;
  - Quais parâmetros existem para cada função;
  - Etc.
- Garantem que as ações mais comuns são amplamente testadas em cada parte do *software;*
- É impossível testar todas as variações;
- Teste interno.

Testes de Caixa Preta

- Pouca ou nenhuma informação sobre o funcionamento interno do jogo:
  - Build *beta* de um jogo num console de testes.
- Teste interno ou externo.

Testes de Caixa Branca

- Usa funções internas ou uma ferramenta de *software* (normalmente um *debugger*) para encontrar erros durante a execução do código;
- Teste interno.

Testes Abertos - Dicas Gerais

- Foco na experiência;
- Pessoas que não participam do desenvolvimento:
  - Amigos e família tendem a sempre elogiar.
- **Não tenha medo de deixar outros jogarem.**

Testes Abertos - Dicas Gerais

- Jogadores muito experientes:
  - Tendem a ser mais rigorosos e enviesados.
- Jogadores pouco experientes:
  - Tendem a ser menos enviesados;
  - Mais desejáveis para testar.
- Jogadores de fora do público-alvo:
  - Sugestões de elementos de outros tipos de jogos que podem fazer parte do seu jogo e deixar mais interessante.
- Variedade é importante.

Testes Abertos - Dicas Gerais

- O que testar?
  - Experiência ao baixar, executar, e/ou registrar/criar conta;
  - Experiência do tutorial, passo a passo e outras dicas/guias;
  - No primeiro contato do jogador com o jogo, ele entende os controles, interfaces, mecânicas e sistema de níveis e de recompensas?
  - Os jogadores desistem do jogo? Em que momento e por quê?

Testes Abertos - Dicas Gerais

- Fale com os testadores o mínimo possível;
- Não rejeite nenhum *feedback*:
  - Mesmo que você já planeje fazer isso;
  - Mesmo que você já saiba que está errado;
  - Deixe a pessoa falar.
- Você pode achar que o usuário vai falar de um problema que já conhece, mas pode ser algo diferente.

Testes Abertos - Dicas Gerais

- Ordem do *feedback* é importante!
  - Mostra o que é mais importante para eles;
  - Onde encontraram os problemas apresentados.
- Pode ser que o elemento que você acha mais engajante do seu jogo não é o mesmo que os jogadores percebem.

Testes Abertos - Dicas Gerais

- Experiência de um jogo é sobre emoções;
- Observe as reações dos jogadores;
- Veja onde eles gastam mais tempo olhando o jogo;
- Fazer um questionário pode ajudar:
  - Deixe espaço para comentários no fim.

Testes Abertos - Dicas Gerais

- Evite ouvir apenas conclusões do tipo:
  - “Faltam powerups”;
  - “Está muito difícil”.
- Pergunte mais para entender as causas desses problemas;
- Deixe os testadores livres para desistir do jogo quando quiser.

Testes Abertos - Dicas Gerais

- Sempre procure saber como o jogador se sente ao jogar;
- Aceite palavras de valor*, mas vá adiante:
  - “bom”, “ruim”, “gostei”, “não gostei”.
- Exemplos:
  - “O que você sentiu enquanto jogava?”
  - “Quando você se sentiu assim?/O que te fez sentir isso?”
  - “Que parte do jogo mais te fez sentir assim?”
  - “Algo que os outros jogadores/o jogo fez aumentou esse sentimento?”

Coleta de Dados Dentro do Jogo

Coleta de Dados Dentro do Jogo

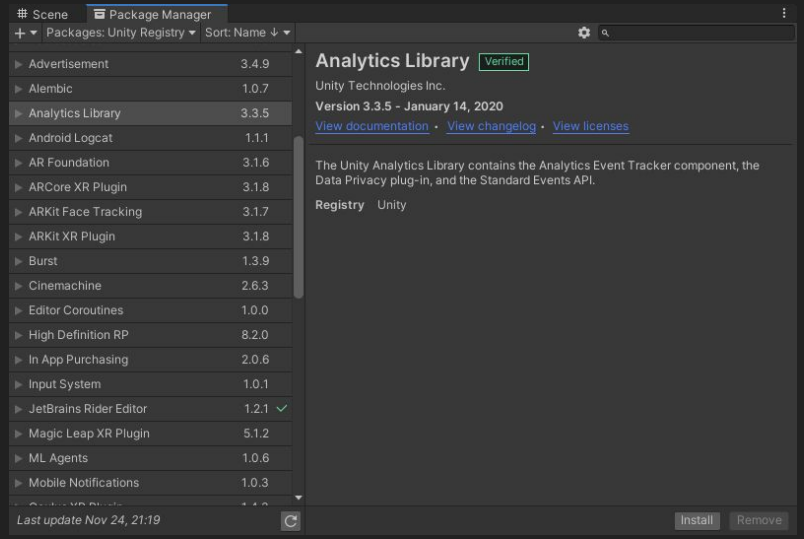
- Todas os dados de ações que o jogador realiza no sistema;
- Não revelar informações sobre os usuários;
- Informar usuários sobre a coleta de dados;
- Status do Jogador em momentos chave;
- Tempo para completar uma seção.

Coleta de Dados Dentro do Jogo

- Quantas vezes o jogador morreu/perdeu;
- Quantas fases o jogador completou/desistiu;
- Ações por segundo;
- *Heatmap*.

Unity Analytics

- Pacote *Analytics Library*;
- Instalação via *Package Manager > Unity Registry*;
- A Unity coleta e envia dados para um Unity Web Service (https://analytics.cloud.unity3d.com/).



Unity Analytics

- Unity dispara eventos quando alguém usa o jogo;
- Alguns eventos são disparados automaticamente;
  - *Core Events;*
  - Eventos baseados em seção e dispositivo.
- Outros são padronizados e acompanham experiência de usuário e comportamento do jogador (*Standard*);
- Você também pode criar eventos customizados (*Custom*).

Unity Analytics

- Grande problema: limitação.
  - **100 eventos por hora (por instância do jogo);**
  - 10 parâmetros por evento;
  - 500 bytes de caracteres de dados enviados por evento;
  - 100 caracteres no nome de um evento customizado;
  - Dashboard mostra os 5000 eventos, parâmetros e combinações de valores de parâmetros mais frequentes em um determinado dia.

Unity Analytics

- Standard: AnalyticsEvent.**NomeDoEvento**();
  - AnalyticsEvent.GameStart();
  - AnalyticsEvent.LevelStart(levelIndex);
  - AnalyticsEvent.GameOver();
  - AnalyticsEvent.LevelComplete(levelIndex, customParams);
  - AnalyticsEvent.LevelFail(levelIndex, customParams);
  - AnalyticsEvent.LevelSkip(levelIndex, customParams);
  - AnalyticsEvent.LevelQuit(levelIndex, customParams);

Unity Analytics

- CustomParams é um Dicionário:

```
Dictionary<string, object> customParams = new Dictionary<string, object>();
customParams.Add("seconds_played", secondsElapsed);
customParams.Add("keys", Player.instance.keys.Count);
customParams.Add("locks", Player.instance.usedKeys.Count);
```

Bug Fixes

Bug Fixes - Bugs Comuns

- Usar Rigidbody e Character Controller juntos:
  - Character Controller foi projetado para controle clássicos, por exemplo plataforma ou FPS;
  - Ridigbody é um componente não-deformável afetado pela gravidade e outras forças físicas;
  - Solução: escolha a opção que faça mais sentido pro seu jogo.

Bug Fixes - Bugs Comuns

- Modificar diretamente o Transform do Rigidbody:
  - Não é uma boa prática mudar a posição ou rotação de um Rigidbody;
  - Ao invés disso, utilize os métodos:
    - AddForce(): para mudar posição;
    - AddTorque(): para mudar rotação.
  - Atribuir a posição de um Rigidbody é válido apenas ao:
    - Adicionar objetos na cena pela primeira vez (spawn);
    - Resetar a cena.

Bug Fixes - Bugs Comuns

- Rigidbody afundando parcialmente em outros objetos:
  - Game engines normalmente possuem um valor de erro que define o contato de objetos;
  - Na Unity, é chamado de Default Contact Offset e possui 0.01 como valor padrão;
  - Para deixar esse controle mais preciso, basta diminuir esse valor.

Bug Fixes - Bugs Comuns

- Objetos rolando para sempre:
  - Simulação da resistência de rolagem não é precisa;
  - Solução: mudar valor de Angular Drag (padrão: 1.0);
  - O valor ideal depende do comportamento desejado.

Bug Fixes - Bugs Comuns

- Mais bugs comuns e mais detalhes em: https://gamedevelopment.tutsplus.com/articles/how-to-fix-common-physics-problems-in-your-game--cms-21418.

Como Facilitar o Debug?

- Uma instrução por linha;
  - Instruções curtas (80 ou 120 caracteres de largura);

```
// Evitar
return ShowMatchSummaryWindow(GetWinner().Name, GetLoser().Name,
Time.ToSeconds(GetMatchTime())).Owner == GetHost();

// Fazer
string winnerName = GetWinner().Name;
string loserName = GetLoser().Name;
int matchTimeInMilliseconds = GetMatchTime();
float matchTimeInSeconds = Time.ToSeconds(matchTimeInMilliseconds);
Player windowOwner = ShowMatchSummaryWindow(winnerName, loserName,
    matchTimeInSeconds);

bool windowOwnerIsHost = windowOwner == GetHost();
return windowOwnerIsHost;

// Fonte:
http://gamesarchitecture.com/debugging-part-i-7-good-code-practices-that-will-help-you-i
n-debugging/
```

Como Facilitar o Debug?

- Transformar operações lógicas complexas em variáveis ou métodos separados;

```
// Evitar
public void AttackPlayer(Player player)
{
    if (GetDistanceTo(player) <= rangeOfView && player.HasCamouflageActive() &&
      !Physics.Raycast(this, player, rangeOfView))
    {
        // attack the player
    }
}


// Fonte:
http://gamesarchitecture.com/debugging-part-i-7-good-code-practices-that-will-help-you-i
n-debugging/
```

Como Facilitar o Debug?

- Operações lógicas complexas em variáveis ou métodos separados;

```
// Fazer
public void AttackPlayer(Player player)
{
    if (IsPlayerVisible(player)) { /* attack the player */ }
}

private bool IsPlayerVisible(Player player)
{
    bool isInRangeOfView = GetDistanceTo(player) <= rangeOfView;
    bool hasCamouflageActive = player.HasCamouflageActive();
    bool isCoveredByObstacle = Physics.Raycast(this, player, rangeOfView);

    return isInRangeOfView && !hasCamouflageActive && !isCoveredByObstacle;
}


// Fonte:
http://gamesarchitecture.com/debugging-part-i-7-good-code-practices-that-will-help-you-in-debugging/
```

Como Facilitar o Debug?

- Programe de forma defensiva (não esconda os bugs!);

```
// Evitar
public void RemoveCondition(ConditionType conditionType)
{
    if (!_initialized) Initialize();
    _conditionsManager.RemoveCondition(conditionType);
}


// Fazer
public void RemoveCondition(ConditionType conditionType)
{
    if (!_initialized) Debug.LogError("Conditions Behaviour is not initialized, so
the conditions won't be applied. Initialize it first.");
    _conditionsManager.RemoveCondition(conditionType);
}



// Fonte:
http://gamesarchitecture.com/debugging-part-i-7-good-code-practices-that-will-help-yo
u-in-debugging/
```

Como Facilitar o Debug?

- Mantenha o tempo de vida das variáveis o menor possível;
  - Evite variáveis globais sempre que puder.
- Mantenha o aninhamento o menor possível;
  - Utilize formatadores automáticos de código: https://stackoverflow.com/questions/49500433/auto-format-c-sharp-code-in-visual-studio-code.
- Escreva comentários.

Referências

- Playtesting - How to Get Good Feedback on Your Game - Extra Credits: https://www.youtube.com/watch?v=on7endO4lPY
- 7 Different Types of Game Testing Techniques: https://dzone.com/articles/7-different-types-of-game-testing-techniques
- Game testing: https://en.wikipedia.org/wiki/Game_testing
- Everything You Need to Know about User Testing Your Game: https://www.usertesting.com/blog/user-testing-games/
- Quality Quality Assurance: A Methodology for Wide-Spectrum Game Testing: http://www.gamasutra.com/view/feature/132398/quality_quality_assurance_a_.php
- Differences between Software Testing and Game Testing: http://www.gamasutra.com/blogs/JohanHoberg/20140721/221444/Differences_between_Software_Testing_and_Game_Testing.php

Referências

- A Simple Way to get Great Playtesting Feedback: http://gamasutra.com/blogs/FilipWiltgren/20160217/265931/A_Simple_Way_to_get_Great_Playtesting_Feedback.php
- A vida de um *QA Tester:* https://kotaku.com/quality-assured-what-it-s-really-like-to-play-games-fo-1720053842
- What Is Games 'User Experience' (UX) and How Does It Help?: https://medium.com/@player_research/what-is-games-user-experience-ux-and-how-does-it-help-ea35ceaa9f05
- Debugging (Part I) – 7 good code practices that will help you in debugging: http://gamesarchitecture.com/debugging-part-i-7-good-code-practices-that-will-help-you-in-debugging/

Referências

- Easy method to fix bugs in your game – Debugging (Part II): http://gamesarchitecture.com/easy-method-to-fix-bugs-in-your-game/
- How to Fix Common Physics Problems in Your Game: https://gamedevelopment.tutsplus.com/articles/how-to-fix-common-physics-problems-in-your-game--cms-21418
- Analytics Library:
  - https://docs.unity3d.com/Manual/UnityAnalyticsEvents.html
  - https://docs.unity3d.com/Manual/UnityAnalyticsCoreEvents.html
  - https://docs.unity3d.com/Manual/UnityAnalyticsStandardEvents.html
  - https://docs.unity3d.com/Manual/UnityAnalyticsCustomEventScripting.html