CRIAÇÃO DE DOCUMENTO DE GAME DESIGN E DEMO DE JOGO UTILIZANDO UNITY

PAULA, Gustavo Tartaglia Silva de[1]; TREVIZANO, Waldir Andrade[2]; BAIA, Joás Weslei[2]; PEREIRA, Ana Amélia de Souza[2]

[1] Ciência da Computação - UNIFAGOC
[2] Ciência da Computação - UNIFAGOC

gustavo.tartaglia2@gmail.com
waldir@unifagoc.edu.br

RESUMO

O processo de desenvolvimento de jogos eletrônicos é complexo está em constante evolução, lidando com diversas decisões que podem mudar o rumo do produto final para um resultado não desejável. Para mitigar esse problema, são utilizadas diversas ferramentas, entre elas o documento de design de jogo. O objetivo deste trabalho é a criação de uma demo de um jogo em Unity, baseado noseu documento de game design.

Palavras-chave: Desenvolvimento de jogos. Design de jogo. Unity.

INTRODUÇÃO

Contando com diversos recursos gratuitos para uso pessoal, como os motores de jogosUnreal Engine, Unity e Godot, a área de desenvolvimento de jogos eletrônicos pode ser explorada por qualquer pessoa que tenha um computador e interesse em código. Além disso, existem plataformas como a Unity Learn, que é disponibilizada gratuitamente pela própria Unity para aprendizado de iniciantes (UNITY s.d.).

Entretanto, mesmo com diversos tutoriais sobre programação no desenvolvimento de jogos, o processo continua sendo árduo, contendo diversas decisões criativas e técnicas importantes que podem mudar o rumo do produto final. Um dos dilemas enfrentados durante a fase de desenvolvimento é a falta de certeza sobre quais elementos do jogo devem ser mantidos, removidos ou adicionados. Segundo Tomandl (2017), essa ambiguidade é algo presente no desenvolvimento da maioria dos jogos, porém, é necessário tomar decisões relacionadas à visão principal do jogo o mais cedo possível e informar a equipe sobre elas.

Por conta disso, algumas ferramentas importantes são utilizadas por times de desenvolvedores hoje em dia, por exemplo, os documentos de design de jogo, para anotar os conceitos principais do jogo, a fim de auxiliar durante outras partes da produção, e ainda os documentos de design técnico, que armazenam todos os planos necessários para os engenheiros desenvolverem o jogo (Bethke, 2003).

O mercado de jogos no Brasil e no mundo tem crescido exponencialmente nos últimos anos, movimentando US $175,8 bilhões em 2021 e com previsão de mais de US $200 bilhões em 2023, segundo Pacete (2022). Por conta disso, as fases de pré-produção são de extrema importância, considerando que elas têm o objetivo de estabelecer uma base forte para a produção (Bramble, 2023), evitando problemas que podem ser mais caros de resolver no futuro.

Sendo assim, o objetivo deste trabalho é criar um documento de game design

de um jogo *beat 'em up* 2D, assim como uma demo desse jogo. Para isso, será necessário: criar documento de design de jogo, detalhando todos os níveis, a história, objetivos do jogo, mecânicas e como elas interagem entre si; pesquisar e usar técnicas utilizadas por profissionais da indústria durante o ciclo de produção de um jogo eletrônico; e desenvolver uma demo jogável por alguns minutos a partir do documento de design de jogo.

REFERENCIAL TEÓRICO

Documento de Design de Jogo

Segundo Bates (2004), o documento de design de jogo deve estar pronto ao final da fase de pré-produção, contendo descrições detalhadas de tudo que irá fazer parte do jogo. Tudo que for detalhado nele será utilizado para o planejamento da produção. O autor citado também afirma que, durante o ciclo de desenvolvimento, esse documento deve sempre ser atualizado para representar o estado mais atual do jogo, e deve conter informações sobre: gameplay, interface, história, personagens, monstros e IA. Também é possível organizar essas informações na forma de páginas em uma Wiki.

Essa documentação tem o objetivo de guiar o time de desenvolvedores durante o processo de desenvolvimento do jogo, mantendo o time inteiro em sincronia do que deve ser feito. Além disso, ela também ajuda a responder a outras dúvidas em questão de planejamento, mantendo o time seguro sobre o que está sendo feito (Bethke, 2003).

Documento de Design Técnico

O objetivo do documento de design técnico é estabelecer os requerimentos técnicos do projeto a partir do que foi descrito no documento de design de jogo. Nele, serão colocadas as ferramentas necessárias para o desenvolvimento, especificando quais serão feitas pelo próprio estúdio e quais serão licenciadas (Bates, 2004).

Esse documento também contém planos de quem lidará com cada requerimento e quando isso deve ser feito. Para isso, ele é dividido em diversos estágios, cada um com sua própria documentação (Bethke, 2003).

Unity

Unity é um motor de jogos 3D e 2D amplamente usado no mercado de desenvolvimento de jogos, sendo utilizado em 70% do top 1000 jogos para dispositivos móveis em 2022 (UNITY, Our Company, s.d.). Além disso, o sistema Unity também conta com suporte para lançamento em múltiplas plataformas como WebGL, desktop, iOS, Android, Playstation, Nintendo Switch e Xbox (UNITY, Multiplatform, s.d.).

O desenvolvimento em Unity utiliza como aspecto fundamental a utilização dos chamados GameObjects. Essas são representações de qualquer objeto em uma cena, como personagens, objetos e terrenos (Unity Manual, Creating a 2D Game, s.d.). Esses GameObjects têm componentes dentro deles que podem ser alterados na própria interface do Unity e que são responsáveis pelo seu comportamento (Unity Manual, Creating a 2D Game, s.d.). Alguns exemplos destes são o *Sprite Renderer,* que renderiza os sprites, os *Colliders,* que são responsáveis pelo formato físico de um objeto, e o

Transform, que determina a localização de um GameObject em uma cena (Unity Manual, Creating a 2D Game, s.d.).

Todos esses componentes são nada mais que representações visuais de classes C# que herdam a classe MonoBehaviour (Unity Manual, Creating a 2D Game, s.d.). O usuário pode alterar o modo que esses componentes interagem utilizando seus próprios scripts (Unity Manual s.d.).

MÉTODO DE DESENVOLVIMENTO

O documento de design de jogo foi feito na ferramenta online Google Docs para facilidade de armazenamento e compartilhamento entre máquinas.

Para o desenvolvimento da demo foi utilizado o motor de jogos Unity por conta do seu suporte para jogos 3D e 2D e das experiências prévias do autor com essa ferramenta. Visual Studio e Visual Studio Code foram utilizados como ambientes de programação.

Por utilizar apenas imagens 2D com estilo pixel art, foi utilizado o programa Aseprite para a criação das animações, cenários e todos os outros elementos de arte visual feitos peloautor.

ASPECTOS DO DESENVOLVIMENTO DO DOCUMENTO DE DESIGN DE JOGO

Primeiro foi feito o documento de design de jogo básico, contendo todas as informações iniciais do jogo, como a ideia principal, identidade visual, mecânicas, personagens e níveis.

Conforme consta no documento de game design anexado a este texto, o objetivo principal foi criar um jogo com gameplay e mecânicas simples de entender, mas que tenham profundidade o bastante para que o usuário se mantenha entretido, tentando melhorar sua habilidade durante a jogatina. Para isso, foi escolhida uma história simples de vingança em que os personagens não falam muito, deixando o jogador preencher as lacunas com sua própria imaginação.

Nos tópicos abaixo estão apenas as especificações de design escolhidas para a demo. A documentação de design do jogo, assim como as sessões abaixo, podem ser encontradas nodocumento de design de jogo.

Enredo

O jogador controla um adolescente delinquente que se encontra no meio de uma
guerra de cinco gangues de escola, tendo como objetivo a sua vingança contra um dos maiores valentões da cidade, por ter feito seu melhor amigo ir parar no hospital. Para vencer essa luta, o nosso protagonista deve ir de gangue em gangue derrotando todos até conseguirsua revanche contra seu rival.

Gameplay

O jogo é do gênero *beat 'em up* 2D de arena, em que o jogador deve derrotar ondas de inimigos com golpes de artes marciais em um local fechado. Após derrotar cem inimigos, um chefe irá aparecer e, caso ele seja derrotado, o próximo nível é liberado. Caso o jogador perca toda sua barra de vida, ele é derrotado.

O personagem principal pode andar para todas as direções, bloquear, desviar e atacar.Durante o bloqueio, ele nega o dano recebido de qualquer direção, porém cada ataque recebido durante esse estado irá desgastar um pouco da barra de histamina, deixando de negar o dano caso ela acabe. O desvio ou arrancada também tem uma forma na qual o protagonista consegue passar pelos inimigos para sair de situações perigosas, entretanto essa forma de desvio só é possível após alguns segundos sem utilizá-la.

Os inimigos utilizam uma inteligência artificial simples, em que eles sabem a localização do jogador o tempo todo e apenas andam para cima dele, parando para atacar caso o jogadoresteja dentro do alcance do ataque.

ASPECTOS DO DESENVOLVIMENTO DA DEMO

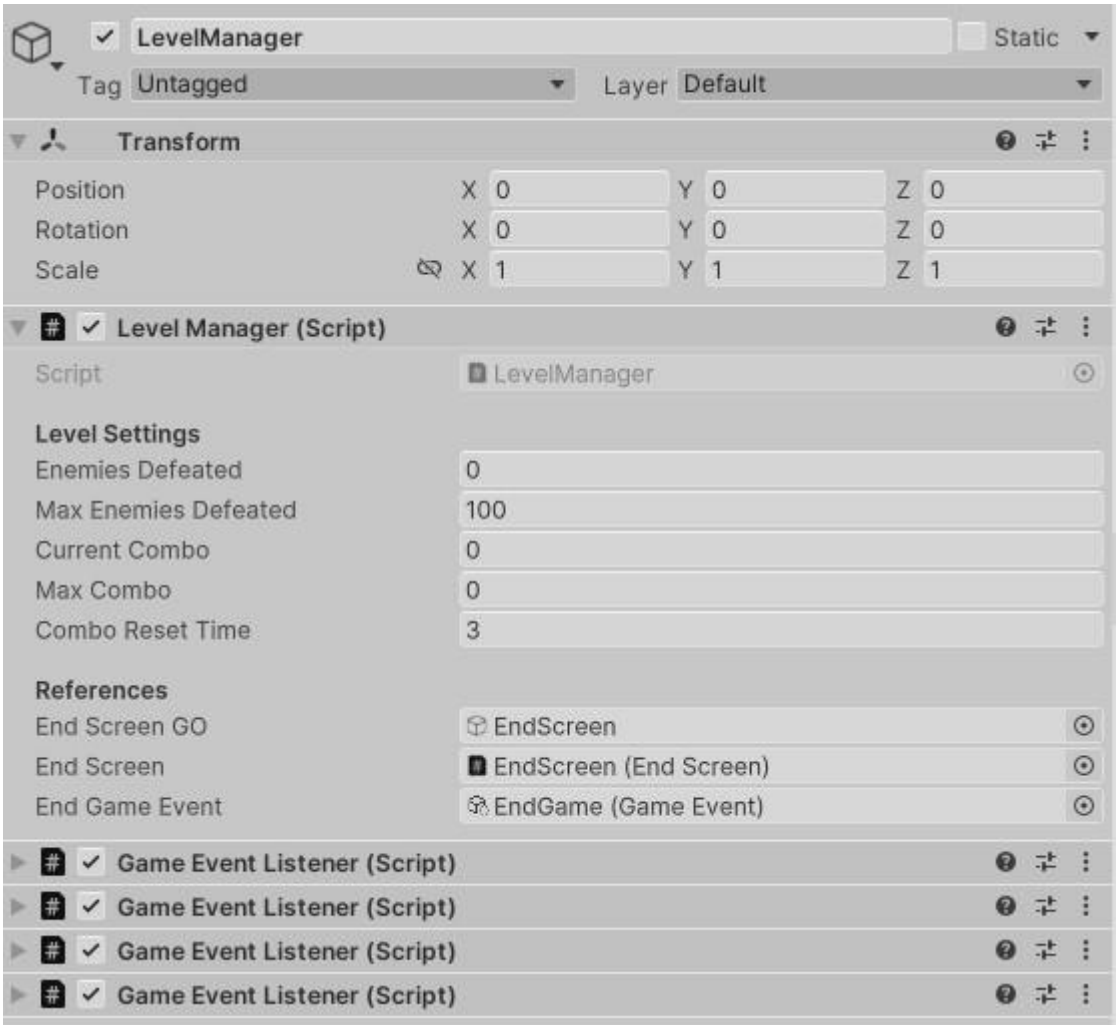
Depois foi criada uma build inicial da demo do jogo em constante evolução e implementando o que foi citado no documento. Por se tratar de uma demo, não estão inclusos todos os aspectos da história ou partesque não são tanto o foco, como diálogos. O mais importante foi demonstrar como funciona o aspecto principal do jogo: o combate.

Para isso, nessa demo é possível jogar apenas um nível de teste, no qual o jogador deve derrotar os cem primeiros inimigos, sendo estes apenas os inimigos base citados no documento de design de jogo. Além disso, o jogador tem acesso aos comandos de ataque, defesa, arrancada e movimentação, como as mecânicas de histamina e arrancada poderosa. Com isso, o usuário pode ter uma pequena experiência do combate do jogo.

Level Manager

O gerenciamento do nível do jogo é feito por um Game Object chamado “Level Manager”, que contém um script de uma classe com o mesmo nome. Esse script tem o objetivo de armazenar informações pertinentes ao nível atual como a quantidade de inimigos derrotada, a quantidade máxima que deve ser derrotada e o nível de combo do jogador, além de executar a lógica necessária ao acontecer alguma situação que leve ao fim do jogo.

Figura 1 – Visão da classe "Level Manager" pelo inspetor do Unity



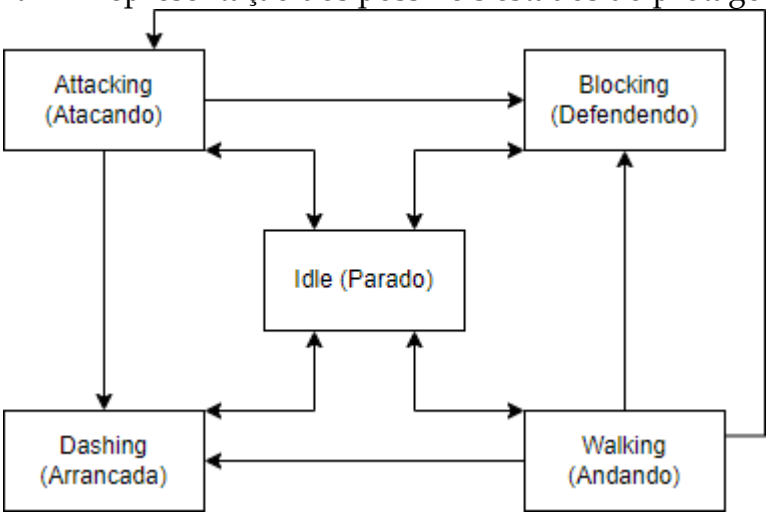
Fonte: elaborada pelos autores.

As funções que executam a tela de vitória ou derrota estão nesse script e são chamadas por eventos que são acionados pelo jogador ou pelos inimigos.

O Protagonista do Jogo

O personagem do jogador deve apresentar comportamentos e ações diferentes dependendo do seu estado atual, por isso foi utilizado o modelo de uma máquina de estadossimples.

Figura 2 – Representação dos possíveis estados do protagonista



Fonte: elaborada pelos autores.

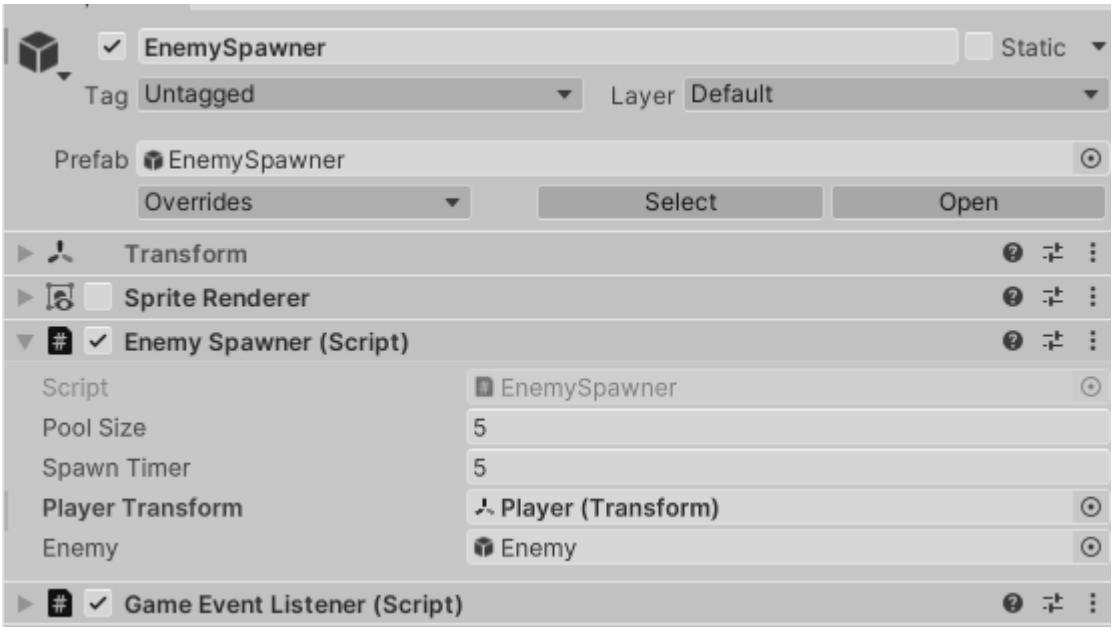
Para isso foi criada uma classe abstrata que serve de base para todos os estados chamada PlayerBaseState. Nessa classe existem funções que executam ações na entrada e saída de um estado e também durante as chamadas do Update e FixedUpdate do Unity. Além disso, também foi criada a classe PlayerStateFactory que serve como a fábrica desses estados, gerando estados caso seja necessário durante a troca de um para o outro. Por fim, tudo isso é administrado pela classe PlayerStateManager, que foi criada para manter a informação de qual é o estado atual do jogador e executar as funções do mesmo.

Invocador de Inimigos

O nível tem alguns GameObjects espalhados que são os responsáveis por invocar os inimigos. Para isso é utilizado um processo de pooling, onde todos os inimigos são instanciados assim que o nível é carregado, porém, ficam desabilitados até precisarem ser utilizados. Esse processo limita a quantidade de inimigos que podem ser instanciados ao mesmo tempo, porém, evita ter o processamento extra de ter que instanciar e destruir GameObjects várias vezes.

Cada invocador tem uma classe chamada EnemySpawner e invocam inimigos em intervalos fixos e param de invocar caso o um evento de fim de jogo seja chamado.

Figura 3 – Representação da classe "Enemy Spawner" no inspetor do Unity

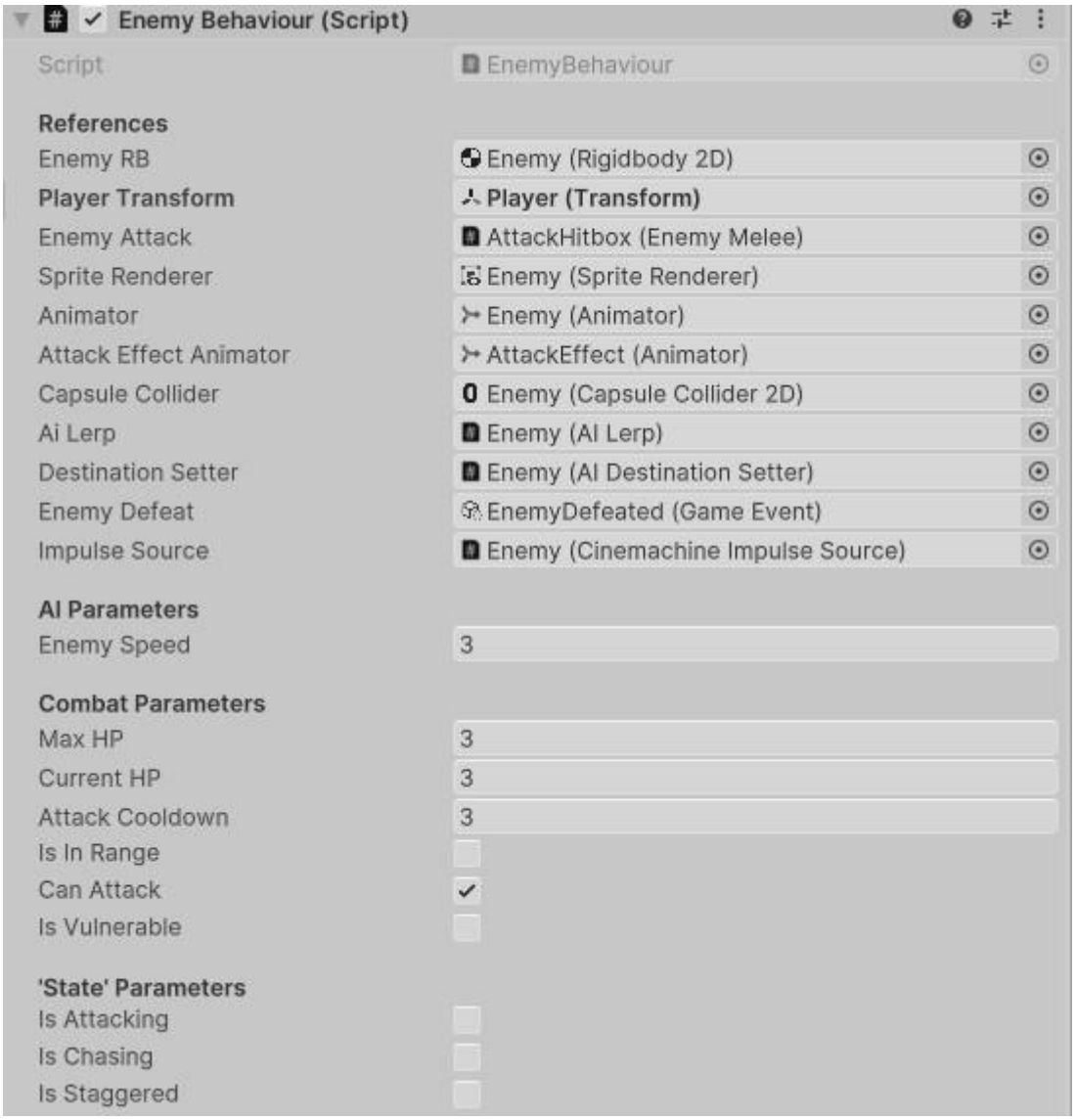


Fonte: elaborada pelos autores.

Inimigos

Os inimigos utilizam um asset chamado A* Pathfinding Project para encontrar o melhor caminho até o protagonista para persegui-lo. Todo o comportamento de cada inimigoé controlado por uma classe chamada EnemyBehaviour, que está presente em cada instânciade inimigo. O GameObject do inimigo contém um componente de Circle Collider 2D que, ao entrar em contato com o jogador, dispara uma notificação para a classe Enemy Behaviour que faz com que o inimigo pare e comece a atacar.

Figura 4 – Representação da classe "Enemy Behaviour" no inspetor do Unity



Fonte: elaborada pelos autores.

A classe que realiza o ataque do inimigo varia dependendo do tipo de inimigo, mas todas contêm a interface IEnemyAttack. Quando o inimigo deve atacar, a classe EnemyBehaviou apenas chama o método da IEnemyAttack chamado Attack para realizar o ataque.

CONSIDERAÇÕES FINAIS

Durante o desenvolvimento deste trabalho o documento de design de jogo sofreu diversas alterações, sendo atualizado de acordo com a ideia original do projeto e sofrendo alterações antes delas serem aplicadas no desenvolvimento da demo, fazendo com que estas

fossem analisadas antes de serem implementadas no projeto. Isto em conjunto com sua habilidade de ser um guia durante a criação da demo fizeram com que o documento de design de jogo cumprisse seu papel de agilizar o processo de desenvolvimento e manter o projeto dentro do escopo original proposto. Pode se dizer que o desenvolvimento da demo sem a utilização desse documento poderia levar à um produto final sem todas as especificações apresentadas, por não ter todos os detalhes especificados.

A criação da demo em si apesar de ter sido difícil, teve seu desenvolvimento facilitadopela utilização do motor de jogos Unity, que simplificou diversos processos e automatizou outros, fazendo com que o foco durante o projeto pudesse ser redirecionado para outras partes também importantes como a criação do documento de design de jogo e das artes utilizadas.

REFERÊNCIAS

BATES, Bob. **Game design**. 2. ed. Thomson Course Technology, 2004. 350 p.

BETHKE, Erik. **Game development and production**. Texas: Wordware Publishing, 2003. 412 p.

BRAMBLE, Ross. **The seven stages of game development**. 4 jan. 2023. Disponível em: https://gamemaker.io/pt-BR/blog/stages-of-game-development. Acesso em: 20 maio 2023.

PACETE, Luiz Gustavo. **2022 promissor:** mercado de games ultrapassará US$ 200 bi até 2023. 3 jan 2022. Disponível em: https://forbes.com.br/forbes-tech/2022/01/com-2022-decisivo-mercado-de-games-ultrapassara-us-200-bi-ate-2023/. Acesso em: 20 maio 2023.

TOMANDL, Ruth. Embracing ambiguity: how to do good work when you don't know what to do. **GDC**, 2017. 1 vídeo (1 hora). Disponível em: https://www.youtube.com/watch?v=4DWdnoLosZ8. Acesso em: 20 maio 2023.

UNITY TECHNOLOGIES. **Creating a 2D game**. Disponível em: https://docs.unity3d.com/Manual/Quickstart2DCreate.html#Fundamentals. Acesso em: 20 mai.o2023.

UNITY TECHNOLOGIES. **Multiplatform**. Disponível em: https://unity.com/solutions/multiplatform. Acessoem: 20 maio 2023.

UNITY TECHNOLOGIES. **Our Company**. Disponível em: https://unity.com/our-company. Acesso em: 20 maio 2023.

UNITY TECHNOLOGIES. **Welcome to Unity Learn**. Disponível em: https://learn.unity.com. Acesso em: 20 maio 2023.

ANEXO - DOCUMENTO DE DESIGN DO JOGO

Projeto Fight

4 de setembro de 2023

1. VISÃO GERAL

Projeto Fight é um jogo beat 'em up 2D de arena com elementos de roguelike. Nesse jogo o objetivo do jogador é sobreviver diversas lutas de rua enquanto derrota todos que entram no seu caminho. Tem forte inspiração na estética de delinquente japonês dos anos 80.

2. CONCEITO

O objetivo do projeto Fight é proporcionar um jogo com foco em re-jogabilidade, onde a diversão está na ideia de re-jogar os níveis repetidamente para o jogador melhorar sua execução, conseguindo mais pontos.
O jogo usa o estilo de pixel art 16 bits por conta de simplicidade e estilo.

3. ENREDO

Contexto

O jogo se passa no Japão nos anos 80, durante a formação das gangues de delinquentes.

Estrutura do Jogo

O jogador pode explorar 5 arenas, sendo cada uma em uma escola diferente e representando ambientes escolares diferentes. Essas arenas não são muito grandes e podem apresentar diversos obstáculos

Ação

O jogador pode se movimentar para qualquer direção e realizar golpes à sua frente. Seu objetivo é sempre derrotar todos os inimigos em uma arena e o chefão final daquela arena.

O Protagonista

O protagonista do jogo é a mistura de um estereótipo de delinquente com um herói de filme de ação. Ele é calado e estóico na maior parte do tempo, porém, perde o controle quando as pessoas que ele ama estão em perigo.

4. GAMEPLAY

Visão Geral

A gameplay consiste em diversas lutas em arenas onde o jogador deve utilizar artes marciais para derrotar cem inimigos. Após atingir esse objetivo ele deve enfrentar o chefão final daquela arena para liberar a próxima.

Seleção de Mapa

Após começar o jogo o jogador pode selecionar através de um menu qual das áreas que elejá desbloqueou ele quer jogar. Essa seleção mostra ao jogador qual foi sua pontuação mais alta e informações específicas da área como quais inimigos estarão presentes e quem é o chefe final. Novas áreas são desbloqueadas de forma linear após vencer a anterior.

Combate

Durante as lutas o protagonista pode realizar diversas ações.

- **Movimentação:** O protagonista pode andar para qualquer direção, não conseguindo atravessar inimigos e obstáculos específicos de cada mapa.
- **Ataque:** É possível realizar ataques em sequência que causam dano a todos osinimigos à sua frente caso eles estejam perto o bastante.
- **Arrancada:** A arrancada faz com que o personagem avance rapidamente na direçãoatual da movimentação do personagem. Caso o jogador não esteja dando uma direção, o personagem irá arrancar para frente.
- **Arrancada Poderosa:** Uma versão mais poderosa da arrancada em que o jogador consegue atravessar inimigos e diversos obstáculos durante sua duração, sendo ótima para sair de situações perigosas e se reposicionar. Após ser utilizada, essa versão daarrancada inicia um timer onde só se torna disponível novamente quando esse timer chegar no 0. Essa versão da arrancada é utilizada automaticamente no lugar da arrancada normal caso esteja disponível.
- **Bloqueio:** Quando ativo, o personagem não recebe dano de qualquer direção casotenha histamina o suficiente. Histamina é perdida ao defender ataques e é recarregada automaticamente ao ficar algum tempo sem utilizar a habilidade de bloqueio.

Controles

O jogador consegue controlar a movimentação do protagonista e a navegação nos menuscom as setas do teclado ou as teclas WASD. Ele também pode atacar e confirmar a seleçãonos menus com o botão esquerdo do mouse.

GAMEPLAY	
BOTÃO	**AÇÃO**
WASD, Setas (Teclado); Analógico Esquerdo, Pad (Gamepad)	Movimentação
Botão Esquerdo do Mouse (Teclado); Botão Oeste (Gamepad)	Ataque
Botão Direito do Mouse (Teclado); Botão Sul (Gamepad)	Arrancada
Espaço (Teclado); Gatilho Direito (Gamepad)	Defesa
Esc (Teclado); Botão Start (Gamepad)	Pausar

NAVEGAÇÃO DE MENUS	
BOTÃO	**AÇÃO**
WASD, Setas, Mouse (Teclado); Analógico Esquerdo, Pad (Gamepad)	Navegar Pelo Menu
Enter, Botão Esquerdo do Mouse (Teclado); Botão Sul (Gamepad)	Confirmar Seleção
Esc (Teclado); Botão Leste (Gamepad)	Cancelar Seleção / Voltar

5. MAPAS

Cada arena tem tamanho o bastante para que o jogador possa andar e desviar de inimigos, porém, não é grande o bastante para que ele consiga ficar correndo para sempre. Os inimigos sempre irão aparecer nos pontos mais ao norte, sul, leste e oeste de cada arena.

Estacionamento

O primeiro mapa é o estacionamento da primeira escola e é a arena mais simples de todas.Aqui existem como obstáculos apenas alguns carros que estão estacionados.
Os inimigos aparecem no centro das paredes mais ao norte, sul, leste e oeste do mapa.

6. INIMIGOS

Os inimigos podem se mover para qualquer direção, desde que não seja através de paredesou obstáculos e sempre vão atrás do jogador tentando cercá-lo. Eles se movem até chegar perto o bastante para realizar um ataque e ficam parados na mesma posição até que o jogador saia do seu alcance.
Apesar de poderem ter visuais diferentes dependendo do mapa, o comportamento deles éigual. Esses são os possíveis inimigos que cada mapa pode ter:

- **Padrão:** Tem ataque corpo a corpo e apenas persegue o jogador.
- **Atirador:** Vai atrás do jogador e mantém uma certa distância. De vez em quando joga pedras nele.
- **Corredor:** É lento na maior parte do tempo, porém, consegue carregar uma investida que pode pegar o jogador de surpresa.
- **Tanque:** Tem muita vida, é lento e sempre persegue o jogador. De tempos em tempos carrega um forte ataque que causa dano numa área à sua volta.