UNIVERSIDADE SÃO JUDAS TADEU

COMPUTAÇÃO GRÁFICA E REALIDADE VIRTUAL

A3

ANTONIO BRUNO FIGUEIREDO DE OLIVEIRA | RA: 822165499

CAIO VIEIRA DE CAMARGO | RA: 8222241862

JULIANA MAGIERO BENEDETTI | RA: 822229587

NATÁLIA BREDIKS MILTUS MARQUES | RA: 822226393

PEDRO LEONARDO RODRIGUES FERREIRA | RA: 822167374

RUAN VINICIUS LUZ DOS SANTOS | RA: 8222244748

BUTANTÃ

2024

SUMÁRIO

1. INTRODUÇÃO……..3

2. DESENVOLVIMENTO……...……..5

2.1. CONCEPÇÃO DA IDEIA……...5

2.2. DESIGN GRÁFICO E ÁUDIO...……6

2.3. MECÂNICA DO JOGO E PROGRESSÃO DOS NÍVEIS..…….10

3. IMPLEMENTAÇÃO…….…...…….11

3.1. ENGINE UTILIZADA……11

3.2. LINGUAGEM DE PROGRAMAÇÃO……...14

3.3. CÓDIGOS UTILIZADOS...……15

3.4. PROBLEMAS ENCONTRADOS E SOLUÇÕES.……15

4. LIÇÕES APRENDIDAS……28

5. CRONOGRAMA DE DESENVOLVIMENTO……28

6. CONCLUSÃO……..31

7. REFERÊNCIAS BIBLIOGRÁFICAS……..……...32

1. INTRODUÇÃO

"Get it Garbage" é um jogo 2D que mergulha os jogadores em um mundo devastado, onde a natureza foi devastada e a tecnologia avançou descontroladamente. Neste cenário, o jogador assume o controle de Garby, uma simpática robô inspirado na personagem Eva do filme Wall-E, cuja missão é coletar lixo e enfrentar desafios para restaurar a beleza e a harmonia do ambiente.

Com mecânicas de jogo simples e uma narrativa divertida e desafiadora, o jogo oferece uma jornada emocionante, onde cada fase apresenta novos obstáculos e desafios a serem superados.

Este trabalho descreve o processo de desenvolvimento do jogo "Get it Garbage", desde a concepção da ideia até a implementação final, destacando as escolhas de design, as ferramentas utilizadas e os desafios enfrentados durante o desenvolvimento. Além disso, serão abordadas as lições aprendidas ao longo do processo, oferecendo uma visão abrangente e detalhada do projeto.

2. DESENVOLVIMENTO

Neste tópico, será abordado o processo de desenvolvimento do jogo "Get it Garbage". Serão detalhados aspectos como a concepção da ideia, o design do jogo, a implementação das mecânicas, a criação dos níveis, a integração de gráficos e áudio, os testes realizados e os ajustes feitos ao longo do desenvolvimento. Além disso, serão discutidas as lições aprendidas durante o processo e a contribuição de cada membro do grupo para a realização do projeto.

2.1. CONCEPÇÃO DA IDEIA

A inspiração para o jogo "Get it Garbage" surgiu de forma inesperada, após uma aula sobre criação de personagens pixelados utilizando o website piskelapp. Durante essa aula, enquanto criavam o próprio sprite, os membros do grupo comentaram que o personagem lembrava a Eva, do filme Wall-E. Esse comentário desencadeou uma discussão animada e criativa sobre a possibilidade de desenvolver um jogo focado na coleta de lixo, em um cenário de mundo devastado pela poluição e avanços tecnológicos descontrolados.

Nos encontros seguintes, o grupo explorou diversas ideias para as fases e desafios do jogo, buscando equilibrar diversão e conscientização ambiental. A escolha por um gameplay

simples e intuitivo foi uma decisão unânime, visando alcançar um público amplo e proporcionar uma experiência de jogo envolvente e educativa.

2.2. DESIGN GRÁFICO E ÁUDIO

O design gráfico do jogo "Get it Garbage" foi desenvolvido com o objetivo de criar um ambiente visualmente impactante e imersivo. O sprite do personagem Garby foi criado utilizando o aplicativo Piskelapp, com inspirações do Google, resultando em um personagem charmoso e carismático que se destaca no cenário do jogo. Além disso, alguns objetos coletáveis e obstáculos do jogo também foram criados utilizando o Piskelapp, garantindo uma consistência visual em todo o jogo.

As imagens de background, que retratam um mundo devastado pela poluição e avanços tecnológicos descontrolados, foram desenvolvidas utilizando o Copilot Design. Essas imagens ajudam a criar a atmosfera sombria e desafiadora do jogo, contribuindo para a imersão do jogador no universo de "Get it Garbage".

Os tilesets utilizados no jogo foram adquiridos na loja da Unity, especificamente o pack "Pixel Adventure 1". Esses tilesets foram escolhidos por sua qualidade e compatibilidade com o estilo visual do jogo, facilitando a criação dos diferentes cenários e níveis.

Tileset "Pixel Adventure 1"

A trilha sonora do jogo foi escolhida com cuidado para complementar a atmosfera e a jogabilidade. As músicas foram selecionadas no website Suno, com um foco em músicas com estilo 16-bit, downtempo, slow e ethereal. Essas escolhas musicais contribuem para a imersão do jogador no mundo de "Get it Garbage" e ajudam a criar uma experiência auditiva envolvente e emocionante.

2.3. MECÂNICA DO JOGO E PROGRESSÃO DOS NÍVEIS

As mecânicas de jogo de "Get it Garbage" são simples e intuitivas, projetadas para serem acessíveis a jogadores de todas as idades. O jogador controla Garby, que pode se mover para a esquerda e direita, pular sobre plataformas e evitar obstáculos. A principal mecânica do jogo é a coleta de lixo espalhado pelos níveis. Cada fase apresenta diferentes tipos de lixo que Garby deve coletar para progredir e cada uma com um nível crescente de dificuldade. A progressão dos níveis é projetada para desafiar as habilidades do jogador de forma gradual, introduzindo novos obstáculos e mecânicas à medida que avançam no jogo. A seguir, uma descrição das telas e níveis do jogo:

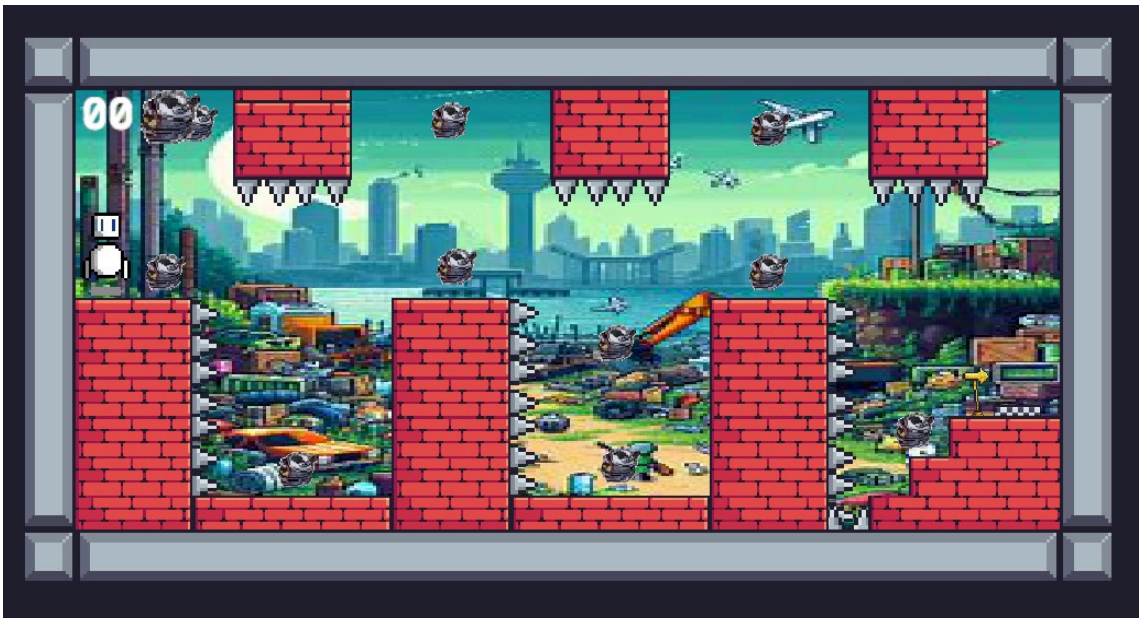
Abertura: A tela de abertura inclui um botão de início, uma breve história e o objetivo do jogo. Esta introdução prepara o jogador para a aventura e estabelece o contexto do mundo devastado que Garby deve salvar.

Nível 1: O jogador deve coletar caixas de suco pulando nas plataformas. Este nível serve como um tutorial básico para familiarizar o jogador com os controles e a mecânica de coleta.

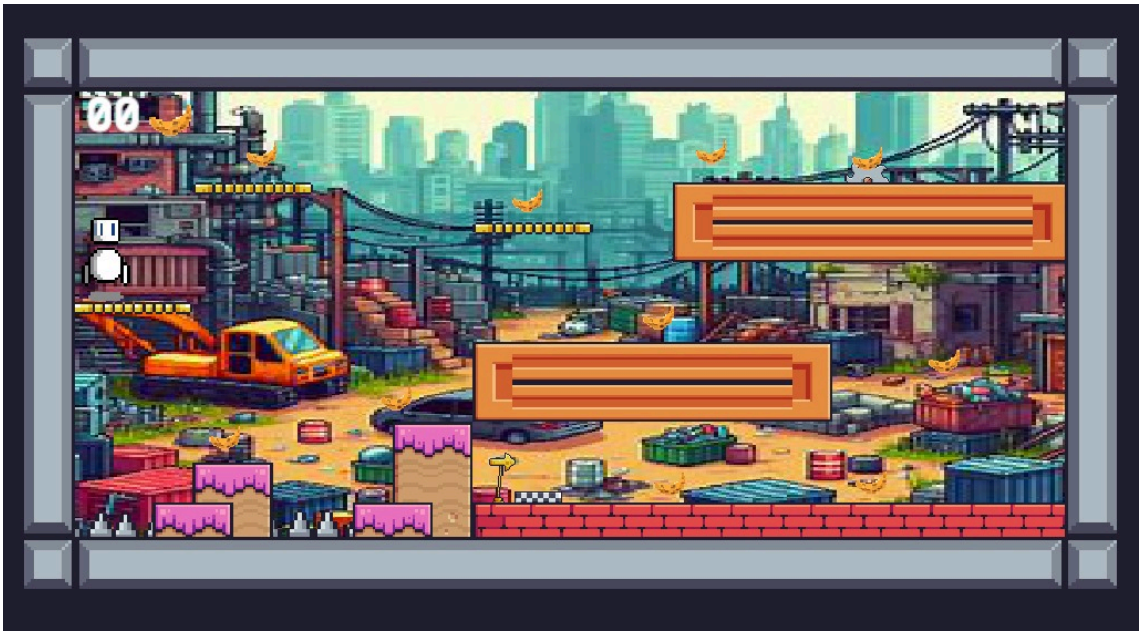
Nível 2: O jogador coleta latinhas pulando nas plataformas e desviando dos espinhos. Este nível introduz os primeiros obstáculos, aumentando a dificuldade.

Nível 3: O jogador coleta tacos estragados ou em decomposição, pulando nas plataformas, desviando dos espinhos e das plataformas que se desmancham/caem. Este nível adiciona uma nova mecânica, exigindo mais precisão nos pulos.

Nível 4: O jogador coleta latinhas, desviando dos espinhos. Este nível mantém a dificuldade consistente com desafios adicionais.



Nível 5: O jogador coleta bananas estragadas, pulando nas plataformas, desviando dos espinhos e das serras que se movem da direita para a esquerda. Este nível introduz serras móveis, aumentando a complexidade dos obstáculos.



Nível 6: O jogador coleta latinhas, pulando nas plataformas, desviando dos espinhos e das serras que se movem da direita para a esquerda. Este nível combina todos os elementos anteriores, proporcionando um desafio final ao jogador.

Fim: A tela final do jogo. Após completar todos os níveis, o jogador é recompensado com uma tela de conclusão, que celebra suas conquistas e encerra a jornada de Garby.

3. IMPLEMENTAÇÃO

3.1. ENGINE UTILIZADA

O jogo "Get it Garbage" foi desenvolvido utilizando a plataforma Unity. A Unity é uma das engines de desenvolvimento de jogos mais populares, conhecida por sua flexibilidade e vasto suporte a diferentes plataformas.

3.2. LINGUAGEM DE PROGRAMAÇÃO

A principal linguagem de programação utilizada foi o C#. Essa escolha se deve à sua compatibilidade com a Unity e à facilidade de uso para a manipulação de objetos e eventos no jogo.

3.3 CÓDIGOS UTILIZADOS

I) Coletáveis

a) Caixa de suco

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Caixinha : MonoBehaviour
{
    private SpriteRenderer sr;
    private CircleCollider2D circle;

    public GameObject collected;
    public int Score;

    void Start()
    {
        sr = GetComponent<SpriteRenderer>();
        circle = GetComponent<CircleCollider2D>();
    }


    void Update()
    {

    }

    void OnTriggerEnter2D(Collider2D collider)
    {
```

```
        if(collider.gameObject.tag == "Player")
        {
            sr.enabled = false;
            circle.enabled = false;
            collected.SetActive(true);

            GameController.instance.totalScore += Score;
            GameController.instance.UpdateScoreText();

            Destroy(gameObject, 0.5f);
        }
    }
}
```

b) Latinha de alumínio #1

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Latinha1 : MonoBehaviour
{
    private SpriteRenderer sr;
    private CircleCollider2D circle;

    public GameObject collected;
    public int Score;

    void Start()
    {
        sr = GetComponent<SpriteRenderer>();
        circle = GetComponent<CircleCollider2D>();
    }

    // Update is called once per frame
    void Update()
```

```
    {

    }

    void OnTriggerEnter2D(Collider2D collider)
    {
        if(collider.gameObject.tag == "Player")
        {
            sr.enabled = false;
            circle.enabled = false;
            collected.SetActive(true);

            GameController.instance.totalScore += Score;
            GameController.instance.UpdateScoreText();

            Destroy(gameObject, 0.5f);
        }
    }
}
```

c) Taco

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Tacco : MonoBehaviour
{
    private SpriteRenderer sr;
    private CircleCollider2D circle;

    public GameObject collected;
    public int Score;


    void Start()
```

```
    {
        sr = GetComponent<SpriteRenderer>();
        circle = GetComponent<CircleCollider2D>();
    }

    void Update()
    {

    }

    void OnTriggerEnter2D(Collider2D collider)
    {
        if(collider.gameObject.tag == "Player")
        {
            sr.enabled = false;
            circle.enabled = false;
            collected.SetActive(true);

            GameController.instance.totalScore += Score;
            GameController.instance.UpdateScoreText();

            Destroy(gameObject, 0.5f);
        }
    }
}
```

d) Latinha de alumínio 2

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Latinha2 : MonoBehaviour
{
    private SpriteRenderer sr;
    private CircleCollider2D circle;
```

```
    public GameObject collected;
    public int Score;

    void Start()
    {
        sr = GetComponent<SpriteRenderer>();
        circle = GetComponent<CircleCollider2D>();
    }


    void Update()
    {

    }

    void OnTriggerEnter2D(Collider2D collider)
    {
        if(collider.gameObject.tag == "Player")
        {
            sr.enabled = false;
            circle.enabled = false;
            collected.SetActive(true);

            GameController.instance.totalScore += Score;
            GameController.instance.UpdateScoreText();

            Destroy(gameObject, 0.5f);
        }
    }
}
```

e) Banana

```
using System.Collections;
using System.Collections.Generic;
```

```
using UnityEngine;

public class Banana : MonoBehaviour
{
    private SpriteRenderer sr;
    private CircleCollider2D circle;

    public GameObject collected;
    public int Score;

    void Start()
    {
        sr = GetComponent<SpriteRenderer>();
        circle = GetComponent<CircleCollider2D>();
    }


    void Update()
    {

    }

     void OnTriggerEnter2D(Collider2D collider)
    {
        if(collider.gameObject.tag == "Player")
        {
            sr.enabled = false;
            circle.enabled = false;
            collected.SetActive(true);

            GameController.instance.totalScore += Score;
            GameController.instance.UpdateScoreText();

            Destroy(gameObject, 0.5f);
        }
```

```
    }
}
```

f) Latinha de alumínio #3

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Latinha3 : MonoBehaviour
{

    private SpriteRenderer sr;
    private CircleCollider2D circle;

    public GameObject collected;
    public int Score;

    void Start()
    {
        sr = GetComponent<SpriteRenderer>();
        circle = GetComponent<CircleCollider2D>();
    }

    void Update()
    {

    }

    void OnTriggerEnter2D(Collider2D collider)
    {
        if(collider.gameObject.tag == "Player")
        {
            sr.enabled = false;
            circle.enabled = false;
            collected.SetActive(true);
```

```
            GameController.instance.totalScore += Score;
            GameController.instance.UpdateScoreText();

            Destroy(gameObject, 0.5f);
        }
    }
}
```

II) Objetos

a) Falling plataform

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class FallingPlataform : MonoBehaviour
{

    public float fallingTime;

    private TargetJoint2D target;
    private BoxCollider2D boxColl;

    void Start()
    {
        target = GetComponent<TargetJoint2D>();
        boxColl = GetComponent<BoxCollider2D>();
    }

    void OnCollisionEnter2D(Collision2D collision)
    {
        if(collision.gameObject.tag == "Player")
        {
            Invoke("Falling", fallingTime);
        }
```

```
    }
    void OnTriggerEnter2D(Collider2D collider)
    {
        if(collider.gameObject.layer == 9)
            {
                Destroy(gameObject);
            }
    }

    void Falling()
    {
        target.enabled = false;
        boxColl.isTrigger = true;
    }

    void Update()
    {

    }
}
```

b) Next level

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class NextLevelPoint : MonoBehaviour
{
    public string lvlName;

    void Start()
    {
```

```
    }


    void Update()
    {

    }

     void OnCollisionEnter2D(Collision2D collision)
    {
        if(collision.gameObject.tag == "Player" )
        {
            // Adicione a condição para verificar se o jogador coletou 10 objetos
            if(GameController.instance.totalScore >= 10)
            {
                SceneManager.LoadScene(lvlName);
            }
            else
            {
                Debug.Log("Você precisa coletar 10 objetos para passar para o próximo nível!");
            }
        }
    }
}
```

c) Serra

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Saw : MonoBehaviour
{
    public float speed;
    public float moveTime;
```

```
    private bool dirRight = true;
    private float timer;

    void Start()
    {

    }


    void Update()
    {
        if(dirRight)
        {
            transform.Translate(Vector2.right * speed * Time.deltaTime);
        }
        else
        {
            transform.Translate(Vector2.left * speed * Time.deltaTime);
        }

        timer += Time.deltaTime;
        if(timer >= moveTime)
        {
            dirRight = !dirRight;
            timer = 0f;
        }

    }
}
```

III) Game

a) Game Controller

```
using System.Collections;
using System.Collections.Generic;
```

```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class GameController : MonoBehaviour
{
    public int totalScore;
    public Text scoreText;

    public GameObject gameOver;

    public static GameController instance;

    void Start()
    {
        instance = this;
    }

     public void UpdateScoreText()
    {
        scoreText.text = totalScore.ToString();
    }

    void Update()
    {

    }

    public void ShowGameOver()
    {
        gameOver.SetActive(true);
    }

    public void RestartGame(string lvlName)
    {
```

```
        SceneManager.LoadScene(lvlName);
    }

}
```

b) Player

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Player : MonoBehaviour
{
    public  float   Speed;
    public  float   JumpForce;

    public  bool    isJumping;
    public  bool    doubleJump;

    private Rigidbody2D rig;
    private Animator anim;



    void Start()
    {
        rig = GetComponent<Rigidbody2D>();
        anim = GetComponent<Animator>();
    }

    void Update()
    {
        Move();
        Jump();
    }

     void Move()
```

```
    {
        Vector3 movement = new Vector3(Input.GetAxis("Horizontal"), 0f,0f);
        transform.position += movement * Time.deltaTime * Speed;

        if(Input.GetAxis("Horizontal") > 0f)
        {
            anim.SetBool("walkl", false);
            anim.SetBool("walkr", true);
        }

         if(Input.GetAxis("Horizontal") < 0f)
        {
            anim.SetBool("walkl", true);
            anim.SetBool("walkr", false);
        }

        if(Input.GetAxis("Horizontal") == 0f)
        {
            anim.SetBool("walkl", false);
            anim.SetBool("walkr", false);

        }
    }

    void Jump()
    {
        if(Input.GetButtonDown("Jump"))
        {
            if(!isJumping)
            {
                rig.AddForce(new Vector2(0f, JumpForce), ForceMode2D.Impulse);
                doubleJump = true;
                anim.SetBool("jump", true);

            }
```

```
        else
        {
            if(doubleJump)
            {
                rig.AddForce(new Vector2(0f, JumpForce), ForceMode2D.Impulse);
                doubleJump = false;
            }
        }

    }
}

void OnCollisionEnter2D(Collision2D collision)
{
    if(collision.gameObject.layer == 8)
    {
        isJumping = false;
        anim.SetBool("jump", false);
    }

    if(collision.gameObject.tag == "Spike")
    {
        GameController.instance.ShowGameOver();
        Destroy(gameObject);
    }

    if(collision.gameObject.tag == "Saw")
    {
        GameController.instance.ShowGameOver();
        Destroy(gameObject);
    }



}
```

```
    void OnCollisionExit2D(Collision2D collision)
    {
        if(collision.gameObject.layer == 8)
        {
            isJumping = true;
        }
    }
}
```

3.4. PROBLEMAS ENCONTRADOS E SOLUÇÕES

Durante o desenvolvimento do jogo "Get it Garbage", encontramos diversos problemas que precisaram ser solucionados para garantir uma jogabilidade fluida e agradável. Abaixo estão alguns dos principais desafios e as soluções implementadas:

1. Aumento da Dificuldade dos Níveis:

- **Problema**: Os níveis não apresentavam uma progressão adequada de dificuldade, tornando o jogo monótono.
- **Solução**: Implementamos novos desafios em cada nível, como plataformas que se desmancham, serras móveis, e espinhos adicionais, para aumentar gradualmente a dificuldade.

2. Troca de Backgrounds:

- **Problema**: Os backgrounds não refletiam corretamente a progressão do jogo e o tema do mundo devastado.
- **Solução**: Utilizamos imagens criadas no Copilot Design para representar diferentes estágios de devastação, tornando a experiência visualmente coesa e imersiva.

3. Ajuste de Fontes:

- **Problema**: As fontes utilizadas inicialmente não eram claras e dificultavam a leitura.
- **Solução**: Selecionamos novas fontes que fossem mais legíveis e adequadas ao estilo visual do jogo.

4. Acumulação de Lixos para Passar de Fase:

- **Problema**: O jogador podia passar de fase sem coletar uma quantidade mínima de lixo, diminuindo o desafio.

- **Solução**: Implementamos uma lógica que exige que o jogador acumule 10 lixos para avançar para o próximo nível, garantindo que o objetivo do jogo seja cumprido.

5. Botão Interativo do Menu:

- **Problema**: O menu inicial não era intuitivo e os botões não estavam responsivos.
- **Solução**: Melhoramos a interface do menu inicial, tornando os botões mais interativos e fáceis de usar.

6. Velocidade da Serra:

- **Problema**: A velocidade da serra era muito alta, impossibilitando a passagem do personagem.
- **Solução**: Ajustamos a velocidade da serra para permitir que os jogadores pudessem passar, mantendo o desafio sem torná-lo impossível.

7. Erro de Código ao Coletar Itens:

- **Problema**: Houve um erro no código que impedia a coleta correta dos itens, afetando a progressão do jogo.
- **Solução**: Revisamos e corrigimos o código responsável pela coleta, garantindo que os itens fossem contabilizados corretamente.

8. Pulo Duplo Infinito:

- **Problema**: O personagem podia executar pulos duplos infinitos, quebrando a mecânica do jogo.
- **Solução**: Ajustamos a lógica de pulo para limitar a quantidade de pulos duplos, garantindo a jogabilidade justa.

9. Espinho Causando Dano Incorretamente:

- **Problema**: Os espinhos davam dano ao personagem mesmo quando ele estava em um bloco ao lado.
- **Solução**: Corrigimos a detecção de colisão dos espinhos para garantir que eles só causassem dano quando o personagem estivesse diretamente em contato.

10. Coletáveis em Locais Inalcançáveis:

- **Problema**: Alguns itens coletáveis estavam em locais que o jogador não conseguia alcançar.

- **Solução**: Reposicionamos esses itens para garantir que todos os coletáveis fossem acessíveis, mantendo o desafio justo.

4. LIÇÕES APRENDIDAS

O desenvolvimento do jogo "Get it Garbage" proporcionou diversas lições importantes, tanto em aspectos técnicos quanto em colaboração de equipe. Entre as principais lições aprendidas estão:

- **Importância de Testes Constantes:** Realizar testes constantes ao longo do desenvolvimento permitiu identificar e corrigir problemas antes que se tornassem críticos. O feedback recebido durante os testes foi essencial para aprimorar a jogabilidade e a experiência do usuário.

- **Iteração e Feedback:** A implementação de mudanças baseadas no feedback dos colegas e dos testes resultou em um produto final mais refinado. Aprendemos que iterar rapidamente e estar aberto a sugestões é fundamental para o sucesso de um projeto de jogo.

- **Gerenciamento de Problemas Técnicos:** Resolvemos diversos problemas técnicos, como o erro de código ao coletar itens, o pulo duplo infinito, e os espinhos que causavam dano incorretamente. Aprendemos a importância de uma abordagem sistemática para depuração e solução de problemas.

- **Ajuste de Mecânicas de Jogo:** Refinar as mecânicas de jogo, como a soma dos coletáveis e a localização de itens, foi crucial para manter o jogo desafiador e justo. Entendemos a necessidade de equilibrar dificuldade e acessibilidade.

5. CRONOGRAMA DE DESENVOLVIMENTO

O cronograma de desenvolvimento do projeto "Get it Garbage" foi cuidadosamente planejado para garantir o progresso eficiente e organizado de todas as etapas de produção. Dividido em semanas, cada uma com tarefas específicas, o cronograma visa cumprir os prazos estabelecidos e as metas de desenvolvimento do jogo. A seguir, detalhamos as atividades planejadas para cada semana, desde a criação do sprite principal até a finalização do documento e dos slides para apresentação.

SEMANA	DATA	TAREFA
SEMANA 1	12/03	Início da criação do sprite principal do jogo no Piskel.
SEMANA 2	19/03	Conclusão do sprite.
SEMANA 3	26/03	Aula de Construct 3 - pausa no projeto.
SEMANA 4	02/04	Aula de Godot - pausa no projeto.
SEMANA 5	09/04	Separação das imagens de coletáveis e criação dos sprites no Piskel.
SEMANA 6	16/04	Início da criação do jogo no Unity - fases 1 e 2 e ambientação.
SEMANA 7	23/04	Inclusão das fases 3, 4, 5 e 6.
SEMANA 8	30/04	Inclusão telas de menu, história/objetivo e fim de jogo.
SEMANA 9	07/05	Semana de prova A1 - pausa no projeto.
SEMANA 10	14/05	Inclusão da OST e finalização do jogo.
SEMANA 11	21/05	Finalização do documento.
SEMANA 12	28/05	Finalização dos slides para apresentação.

6. CONCLUSÃO

O desenvolvimento do jogo "Get it Garbage" foi um processo desafiador e enriquecedor, aplicando conceitos teóricos de Computação Gráfica em um projeto prático. Desde a concepção da ideia até a implementação final, superamos diversos problemas técnicos e de design através de uma abordagem colaborativa.

O principal objetivo era criar uma experiência envolvente em um cenário futurista devastado, onde o jogador coleta lixo enquanto enfrenta desafios crescentes. Utilizando a engine Unity e a linguagem C#, implementamos mecânicas de jogo bem definidas e uma interface intuitiva. As imagens de background, os sprites, e a trilha sonora foram selecionados para criar uma atmosfera coesa e imersiva.

Durante o desenvolvimento, enfrentamos e solucionamos problemas como dificuldade inadequada dos níveis, bugs críticos na coleta de itens, e falhas na detecção de colisão. Estes desafios nos ensinaram a importância de testes contínuos, feedback constante, e uma abordagem sistemática para depuração de código.

As lições aprendidas são valiosas para futuros desenvolvimentos. Compreendemos melhor a necessidade de equilibrar dificuldade e acessibilidade, a importância de documentação clara, e a eficiência da colaboração em equipe. A experiência adquirida será fundamental para enfrentar novos projetos com confiança e competência.

Em resumo, "Get it Garbage" representa um produto final funcional e divertido, além de um crescimento significativo de todos os membros da equipe, atingindo seus objetivos técnicos e proporcionando uma compreensão mais profunda dos desafios e recompensas do desenvolvimento de jogos.

7. REFERÊNCIAS

- https://www.dafont.com/scifly.font
- https://www.dafont.com/pt/press-start-2p.font
- https://www.piskelapp.com/p/create/sprite
- https://assetstore.unity.com/packages/2d/characters/pixel-adventure-1-155360
- https://suno.com/playlist/c56e5bcf-d412-4862-9377-00b818ff71f6