**Licenciatura em E.C.G.M.**

Escola Superior de Tecnologia e Gestão

*Instituto Politécnico de Viana do Castelo*

Unidade curricular de

Sistemas Multimédia

Relatório Final do Trabalho Prático II – Phaser 3.0

2024/2025

João Paulo Martins Novo n25968

ESTG-ECGM 3ºAno

# Objetivo e Funcionalidades do Jogo

História/Objetivo  
  
 No mundo do jogo, os "slimes" eram monstros simpáticos, necessários para o equilíbrio do ecossistema. No entanto, com o uso descontrolado de petróleo, eles foram corrompidos, transformando-se em "slimes de petróleo".

Agora, os slimes corrompidos vagueiam pelo mapa. O jogador tem o papel de salvar a aldeia. Para isso tem que apanhar peças espalhadas pelo mapa para construir uma central de energia éolica.

Durante o jogo, o jogador terá de fugir dos ataques dos slimes, podendo ser atingido no máximo três vezes antes de morrer.

## Funcionalidades Idealizadas

Estarão sinalizadas as funcionalidades concretizadas das idealizadas.

### 1. Apanhar Recursos [X]

* Sistema de apanhar que permite recolher peças espalhadas pelo mapa para construir a central solar.
* Indicação simples da quantidade total no mapa das peças e das que já coletou.
* Conforme a quantidade de recursos coletados poderá ganhar velocidade extra.

### 2. Sistema de Vidas [X]

* Jogador só pode ser atingido no máximo três vezes antes de perder.
* Possibilidade de encontrar vidas extras no mapa.

### 3. Pontuação [X]

* Sistema de pontuação conforme o número de peças recolhidas.

# Protótipo (de Baixa-Fidelidade)

A graph paper with writing on it

Description automatically generated

Fig 1. Protótipo de baixa-fidelidade

# Cumprimento de Requisitos

1. O jogo é original, com o objetivo claro de coletar os itens enquanto o tempo é um fator importante, desafiando o jogador a completar as tarefas rapidamente. Isso adiciona um bom componente de competição para obter o melhor tempo possível.
2. O jogo é subordinado ao tema de “O Futuro da Transição Energética suportado nas Energias Renováveis, devido á contextualização da história e estar a coletar peças da Éolica para reconstrução para livrar os slimes e parar a corrupção devida ao petróleo.
3. O jogo é composto por dois níveis, com mapas distintos. O avanço para o segundo nível é condicionado à conclusão do primeiro, oferecendo uma progressão clara e aumentando a dificuldade conforme o jogador avança.
4. A aleatoriedade na distribuição dos slimes, colecionáveis e corações oferece uma jogabilidade única a cada partida, o que contribui para a rejogabilidade.
5. O jogo tem essencialmente o ecra de Menu (Receção), o ecra das opções, ecrã de como jogar, ecrã do jogo e o ecrã dos resultados.
6. Tanto o player como o slime estão animados e apenas o player é controlado através do teclado e rato.
7. A deteção de colisão é o Arcade Physics.
8. Tanto quanto o click do botão, ao andar do player, coletar os items e corações e por fim o dano quando o slime atinge o jogador.
9. O código está racional conforme as diversas classes necessárias para a devida gestão do jogo.
10. O código está bem estrutura e com todas as funções com a devida nessesariedade comentadas.
11. O Relatório terá as páginas necessárias para a comentação do projeto e desenvolvimento do jogo.

# Requisitos Adicionais

1. O jogo conta com dois mapas realizados no Tilemaps.
2. Tem a possibilidade de controlar se deseja som ou não nas opções.
3. Tem diversas vidas e 2 niveís de jogo.
4. Tem a inclusão de técnica simples de Inteligencia Artificial, mas é uma função do sistema Arcade Physics onde calcula automaticamente a direção e aplica o movimento de forma que o objeto se mova em direção ao alvo.

# Detalhes sobre o Código

## Cena de Jogo

Tanto quanto a cena do jogo 1 e cena do jogo 2 são semelhantes, porque só muda o mapa.

### 1. **Funções de Criação**

* **preload**: Responsável pelo carregamento de recursos, como imagens e sons, antes do início do jogo.
* **create**: Inicializa a cena, configurando a física, o mapa, os obstáculos, o jogador, as animações, os inimigos, os itens colecionáveis e a interface do usuário (UI).
* **criarUI**: Cria a interface do usuário, incluindo corações (vidas), a imagem de energia e o motor eólico.
* **criarTextoInfo**: Exibe as instruções do jogo para o jogador no início.

### 2. **Funções de Movimento e Interação do Jogador**

* **movimento**: Controla o movimento do jogador com base nas teclas pressionadas e ajusta a velocidade com o "boost".
* **fazBoost e tiraBoost**: Ativam e desativam o boost do jogador, exibindo ou ocultando a energia do boost.
* **collisaoInimigo**: Define a colisão entre o jogador e os inimigos (slimes), causando dano e reduzindo as vidas do jogador.
* **perseguirPlayer**: Faz com que os inimigos (slimes) perseguem o jogador, movendo-se em direção à posição dele.

### 3. **Funções de Coleta e Pontuação**

* **createColecao**: Cria os itens colecionáveis (componentes da eólica) espalhados pelo mapa.
* **coletar**: Processa a coleta de itens, aumentando a pontuação e verificando se o jogador coletou todos os itens necessários.
* **createCoracoesVida e coletarVida**: Gerencia a coleta de corações, que aumentam a vida do jogador.

### 4. **Funções de Gerenciamento de Vida e Game Over**

* **perderVida**: Reduz a vida do jogador e verifica se ele perdeu todas as vidas, acionando a função gameOver.
* **ganharVida**: Recupera a vida do jogador, se possível.
* **gameOver**: Finaliza o jogo quando o jogador perde todas as vidas ou atinge um objetivo específico.

### 5. **Funções de Animação e Efeitos Visuais**

* **createAnimacoes**: Define as animações do jogador e dos inimigos (slimes).
* **criarColisaoPlayer**: Cria colisões entre o jogador e obstáculos, inimigos e outros elementos interativos.

### 6. **Funções de Controle de Tempo e UI**

* **atualizaTimer**: Atualiza o contador de tempo no jogo, exibindo-o na tela.
* **atualizarVidas**: Atualiza a UI para refletir a quantidade de vidas restantes do jogador.

## Cena Opções

Nas opções ao criar verifica se o som já está desativado ou não, para que se esteja já mostre o icon de som mutado. Clicando no própria icon muda o estado do som de ligado para desligado e vice-versa.

## Cena GameOver

A cena de Gameover recebe com parâmetro uma string “XX:XXX:X”, onde no primeiro trecho dela vem com um indicativo “Y” ou “N” onde simboliza se recolheu ou não todos os colecionáveis, no trecho a seguir a pontuação e no trecho final diz em que nível está “1” ou “2”.

ex: Y:944:1

# Manual do Jogo

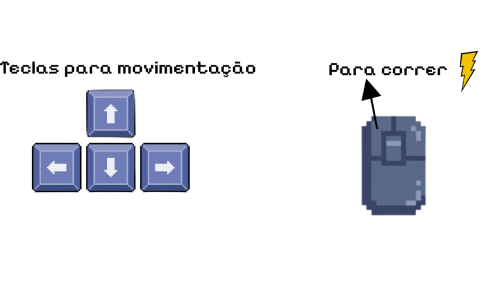
Para se jogar o jogo tem no próprio menu inicial um botão no qual leva para uma cena onde tem instruções de como se jogar o jogo.   
 No qual as instruções são as setinhas para movimento e se premir no click faz com que o jogador “corra”.  


Fig 2. Imagem de Info de gameplay

# Aspetos para Melhoria

Como melhorias para o projeto, algumas mudanças que poderiam ser implementadas são:

1. **Limitação do Boost**: O boost poderia ser restrito a um determinado tempo, seguido de um período de cooldown. Isso impediria de apenas abusar do boost e trazia mais estratégia á mecânica.
2. **Gestão de Níveis na Mesma Cena**: A gestão de níveis poderia ser realizada dentro da mesma cena, uma vez que a transição de mapa é a maior diferença. Isso deixava depois menos classes, mas com diferentes cenas para modificar cada nível e a integração de novas mecânicas diferentes acaba por ser mais facilitado.
3. **Integração de Joystick Virtual**: Implementar um joystick virtual para controlar o personagem, oferecendo a experiência em dispositivos móveis.
4. **Diversificação de Inimigos**: Mais tipos de inimigos conforme os níveis, para maior dificuldade.

No entanto, devido ao tempo limitado e à necessidade de focar em outras partes da entrega, como a realização do relatório e a produção do vídeo promocional. Como também o teste da unidade curricular, essas melhorias não foram possíveis de implementar. O prazo mais apertado e a priorização de outras também tarefas institucionais dificultaram a inclusão dessas funcionalidades extras.