

MODELO PARA O DOCUMENTO DE REQUISITOS DO PROJETO

REQUISITOS FUNCIONAIS

[RF001]		
Nome:	Movimentar o personagem	
Descrição:	O sistema deve permitir que o jogador controle o personagem em diferentes formas de movimento, de acordo com o cenário. Nos cenários da cidade e da floresta, o personagem pode realizar movimentos de pulo. Já no cenário aquático, o personagem pode se mover em todas as direções (direita, esquerda, cima e baixo).	
Atores:	Jogador	
Prioridade:	Alta	
Entradas e pré-condições:		<ul style="list-style-type: none">● O jogo deve estar em execução.● O personagem deve estar ativo em um cenário.
Saídas e pós-condições:		<ul style="list-style-type: none">● O personagem se move conforme os comandos permitidos para o cenário em que se encontra.
Fluxos de eventos		
Fluxo principal:		<ol style="list-style-type: none">1. O jogador pressiona uma tecla de movimento (direita, esquerda, acima, abaixo e pulo).2. O sistema identifica a entrada.3. O personagem se movimenta de acordo com o cenário.
Fluxo secundário 1:		<ol style="list-style-type: none">1. Se a tecla pressionada não for válida, nenhuma ação é realizada e o jogo permanece em andamento.

[RF002]		
Nome:	Coletar lixo no cenário	
Descrição:	O sistema deve permitir que o jogador colete objetos de lixo espalhados pelos cenários.	
Atores:	Jogador	
Prioridade:	Essencial	
Entradas e pré-condições:	<ul style="list-style-type: none">• O personagem deve se movimentar até a posição do objeto de lixo.	
Saídas e pós-condições:	<ul style="list-style-type: none">• O lixo é removido do cenário e adicionado ao inventário do jogador.	
Fluxos de eventos		
Fluxo principal:	<ol style="list-style-type: none">1. O jogador se aproxima do item.2. O sistema detecta colisão com o objeto.3. O item é coletado e removido do cenário.	
Fluxo secundário 1:	-	

[RF003]		
Nome:	Exibir contador de lixo coletado	
Descrição:	O sistema deve exibir, em tempo real, um contador na interface do jogo que apresente a quantidade de lixo coletado pelo jogador, organizado por tipo (ex.: plástico, metal, vidro, papel, orgânico, etc.). O contador deve ser atualizado automaticamente conforme o jogador coleta novos itens.	
Atores:	Jogador	
Prioridade:	Essencial	
Entradas e pré-condições:	<ul style="list-style-type: none">• O jogador deve estar em uma partida ativa.• O jogo deve ter itens de lixo disponíveis no cenário para coleta.	
Saídas e pós-condições:	<ul style="list-style-type: none">• O contador permanece visível e atualizado durante toda a partida.• Os valores de lixo coletado são armazenados até o fim da rodada.	
Fluxos de eventos		
Fluxo principal:	<ol style="list-style-type: none">1. O jogador inicia a partida.2. O jogo posiciona diferentes tipos de lixo no cenário.3. O jogador coleta um item de lixo.4. O sistema identifica o tipo do item coletado.5. O contador em tela é atualizado, incrementando na categoria correspondente ao tipo de lixo6. O jogador pode visualizar em tempo real a quantidade total e por tipo de lixo coletado.	
Fluxo secundário 1:	-	

[RF004]		
Nome:	Desviar de obstáculos	
Descrição:	O sistema deve permitir que o jogador evite obstáculos no cenário (ex.: carros, árvores caídas, pedras).	
Atores:	Jogador	
Prioridade:	Média	
Entradas e pré-condições:	<ul style="list-style-type: none">• O obstáculo deve estar posicionado no cenário.	
Saídas e pós-condições:	<ul style="list-style-type: none">• O jogador passa pelo obstáculo sem colisão.• Caso colida, a fase reinicia.	
Fluxos de eventos		
Fluxo principal:	<ol style="list-style-type: none">1. O jogador se move em direção ao obstáculo.2. O jogador desvia a tempo.3. O jogo continua normalmente.	
Fluxo secundário 1:	<ol style="list-style-type: none">1. O jogador não desvia.2. O sistema aplica penalidade (a fase reinicia).	

[RF005]		
Nome:	Reiniciar cenário após colisão	
Descrição:	O sistema deve reiniciar automaticamente o cenário do jogo sempre que o jogador colidir com um obstáculo ou elemento que represente falha. Dessa forma, a partida é reiniciada sem necessidade de intervenção manual, garantindo a continuidade do jogo.	
Atores:	Jogador	
Prioridade:	Média	
Entradas e pré-condições:		<ul style="list-style-type: none">• O jogador deve estar em uma partida ativa.• O cenário deve conter obstáculos ou elementos que possam causar colisão.
Saídas e pós-condições:		<ul style="list-style-type: none">• O cenário é reiniciado imediatamente após a colisão.• O contador de lixo coletado e a pontuação retornam ao valor inicial.
Fluxos de eventos		
Fluxo principal:		<ol style="list-style-type: none">1. O jogador inicia a partida.2. O jogador percorre o cenário e coleta itens de lixo.3. O jogador colide com um obstáculo.4. O sistema detecta a colisão.5. O sistema reinicia automaticamente o cenário, reposicionando o jogador e os elementos do jogo em seu estado inicial.
Fluxo secundário 1:		<ol style="list-style-type: none">1. O jogador não desvia do obstáculo.2. O jogo continua normalmente.

[RF006]		
Nome:	Classificar lixo nas lixeiras corretas	
Descrição:	O sistema deve permitir que o jogador descarte o lixo coletado em lixeiras específicas (plástico, papel, vidro, metal, orgânico) no final de cada fase.	
Atores:	Jogador	
Prioridade:	Alta	
Entradas e pré-condições:		<ul style="list-style-type: none">• O jogador deve ter lixo coletado no inventário.• Lixeira deve estar disponível no cenário.
Saídas e pós-condições:		<ul style="list-style-type: none">• Lixo é removido do inventário e contabilizado na lixeira correta.
Fluxos de eventos		
Fluxo principal:		<ol style="list-style-type: none">1. O jogador interage com uma lixeira.2. O sistema verifica o tipo de lixo.3. O lixo é corretamente classificado.4. O cenário é atualizado (melhora visual).
Fluxo secundário 1:		<ol style="list-style-type: none">1. O jogador interage com uma lixeira incorreta.2. O sistema detecta a inconsistência.3. O sistema exibe uma mensagem de erro educativa.4. O lixo permanece no inventário do jogador.

[RF007]		
Nome:	Exibir e transformar cenários	
Descrição:	O sistema deve exibir três cenários (cidade, floresta e oceano), que são transformados após a coleta e correta classificação do lixo.	
Atores:	Jogador	
Prioridade:	Essencial	
Entradas e pré-condições:	<ul style="list-style-type: none">O jogador deve completar a coleta e classificação de todos os lixos do cenário atual.	
Saídas e pós-condições:	<ul style="list-style-type: none">O cenário é transformado em sua versão limpa (ex.: cidade sem poluição, floresta com árvores saudáveis, oceano limpo).	
Fluxos de eventos		
Fluxo principal:	<ol style="list-style-type: none">O jogador completa a classificação correta de todos os lixos.O sistema transforma o cenário exibindo sua versão limpa.	
Fluxo secundário 1:	<ol style="list-style-type: none">Se houver lixo não classificado, o cenário não é transformado.	

[RF008]		
Nome:	Avançar para o próximo cenário	
Descrição:	O sistema deve avançar automaticamente para o próximo cenário após a conclusão do anterior.	
Atores:	Jogador	
Prioridade:	Essencial	
Entradas e pré-condições:	<ul style="list-style-type: none">• O cenário atual deve estar limpo e concluído.	
Saídas e pós-condições:	<ul style="list-style-type: none">• O próximo cenário é carregado.	
Fluxos de eventos		
Fluxo principal:	<ol style="list-style-type: none">1. O jogador conclui o cenário atual.2. O sistema carrega o próximo cenário automaticamente.	
Fluxo secundário 1:	<ol style="list-style-type: none">1. Se o cenário não for concluído, o jogo permanece no mesmo estágio.	

[RF009]		
Nome:	Exibir tela de finalização	
Descrição:	O sistema deve exibir uma tela de finalização com mensagem educativa ao término do último cenário.	
Atores:	Jogador	
Prioridade:	Essencial	
Entradas e pré-condições:	<ul style="list-style-type: none">O jogador deve concluir todos os cenários.	
Saídas e pós-condições:	<ul style="list-style-type: none">O sistema apresenta mensagem educativa e encerra o jogo.	
Fluxos de eventos		
Fluxo principal:	<ol style="list-style-type: none">O jogador conclui o último cenário.O sistema exibe tela de finalização com mensagem educativa.	
Fluxo secundário 1:	-	

REQUISITOS NÃO FUNCIONAIS

[RNF01]		
Nome:	Interface gráfica em 2D	
Descrição:	O jogo deve apresentar gráficos bidimensionais (2D).	
Atores:	Jogador	
Prioridade:	Alta	
Entradas e pré-condições:	<ul style="list-style-type: none">● O jogo deve estar em execução.	
Saídas e pós-condições:	<ul style="list-style-type: none">● O cenário é exibido em estilo 2D.	
Fluxos de eventos		
Fluxo principal:	-	
Fluxo secundário 1:	-	

[RNF02]		
Nome:	Usabilidade intuitiva na jogabilidade	
Descrição:	O jogo deve ser simples e de fácil compreensão para jogadores de diferentes idades.	
Atores:	Jogador	
Prioridade:	Alta	
Entradas e pré-condições:		<ul style="list-style-type: none">Jogador deve ter acesso ao jogo.
Saídas e pós-condições:		<ul style="list-style-type: none">Jogador compreende facilmente os comandos e objetivos.
Fluxos de eventos		
Fluxo principal:	-	
Fluxo secundário 1:	-	

[RNF03]		
Nome:	Feedback visual e sonoro para ações do jogador	
Descrição:	O sistema deve fornecer respostas visuais (animações, mudanças gráficas) e sonoras (efeitos de som) às ações do jogador.	
Atores:	Jogador	
Prioridade:	Média	
Entradas e pré-condições:		<ul style="list-style-type: none">O jogador realiza uma ação (movimento, coleta, reciclagem).
Saídas e pós-condições:		<ul style="list-style-type: none">O sistema responde com feedback correspondente.
Fluxos de eventos		
Fluxo principal:	-	
Fluxo secundário 1:	-	

[RNF04]		
Nome:	Mensagens educativas relacionadas aos ODS	
Descrição:	O sistema deve exibir mensagens de conscientização ambiental alinhadas aos Objetivos de Desenvolvimento Sustentável (ODS).	
Atores:	Jogador	
Prioridade:	Média	
Entradas e pré-condições:	<ul style="list-style-type: none">Jogador deve concluir cenários ou ações importantes.	
Saídas e pós-condições:	<ul style="list-style-type: none">Mensagem educativa é apresentada na tela.	
Fluxos de eventos		
Fluxo principal:	-	
Fluxo secundário 1:	-	