Game Design Document

Bird Survivor

Introdução

Bird Survivor é um jogo 2D para computador onde o personagem principal deve sobreviver aos inimigos enquanto anda por um mapa limitado e que usa a perspectiva top-down.

Jogadores: 1 jogador
Tempo estimado da partida: 4 minutos

Objetivo

Sobreviver por mais tempo, acumulando habilidades e lidando com o acúmulo de inimigos que nascem pelo mapa mesmo em posições distantes da área visível pelo jogador.

Controles

O personagem principal é controlado pelas setas do teclado.

Para recomeçar o jogo após o game over é necessário apertar a barra de espaço.

Funcionamento

Mapa

O mapa é uma área retangular, sem obstáculos, maior do que a área visível na tela. O jogador sempre observa uma seção desse mapa, nunca ele inteiro, independente do tamanho da janela ou da resolução.

Inimigos

Os inimigos nascem aleatoriamente pelo mapa e andam na direção do jogador, mesmo que eles estejam posicionados fora da área visível na tela. Ou seja, a área de visão do jogador não é a dos inimigos. A área de visão dos inimigos é “infinita”, diferente da área de visão do jogador explicada anteriormente.

Os inimigos são pássaros roxos com o equivalente a 2 de vida. Ou seja, precisam que 2 projéteis o atinjam para que eles morram. Eles caminham devagar. O objetivo não é pegar o jogador de surpresa, mas sim cercá-lo. Por conta dessa baixa velocidade, se torna difícil agrupar os inimigos em uma área do mapa e correr para outra, uma vez que ao tentar agrupá-los eles vão

demorar tanto para se aglomerar que novos inimigos já terão nascido na parte mais distante do mapa.

Jogador

O jogador atira em intervalos fixos na direção dos inimigos. O tiro vai em linha reta até encontrar um inimigo ou sair do mapa. Inicialmente, ao encontrar um inimigo ele aplicará 1 de dano e desaparecerá. Se o tiro saiu na direção de um inimigo mas ele se moveu para fora da sua trajetória, ele simplesmente não será atingido.

Ao mater um inimigo, o jogador automaticamente ganha 10 de experiência. Ao alcançar 100 de experiência, ele passa de nível. A cada nível a quantidade de experiência adicionada por matar um inimigo cai 5%.

Ao alcançar um novo nível o jogador deve escolher uma habilidade nova para adicionar às habilidades conquistadas. Serão exibidas 3 opções dentre as habilidades, escolhidas aleatoriamente.

Habilidades

As habilidades que podem ser adicionadas pelo jogador são:

- Aumentar a velocidade do tiro;
- Diminuir o intervalo entre tiros;
- Atirar mais de uma vez por intervalo;
- Andar mais rápido.

A habilidade de atirar mais de uma vez por intervalo causa, na prática, um aumento muito grande na capacidade de dano do jogador, por isso sua chance de aparecer dentre as 3 possibilidades é de apenas 10%. As outras habilidades aparecem 30% das vezes.

Implementação

Mapa e Personagens

Como vimos em aula, o mapa foi implementado utilizando um TileMap, e o jogador é um CharacterBody2D. O jogador e os inimigos contêm AnimatedSprite2D com as sprites mapeadas a partir de uma imagem que contém todas as sprites dos personagens andando em diferentes direções.

Câmera

A câmera foi fixada no jogador para que ela o siga num mapa que é muito maior que o tamanho da janela. Ao mesmo tempo, um efeito de smoothness foi aplicado à sua trajetória para que ela não o siga imediatamente. Além dos limites físicos dos personagens serem maiores que os da câmera, como é o esperado, fazendo com que ela pare nas bordas do mapa e não mostre o que há do lado de fora, enquanto o personagem consegue andar até bem mais próximo da borda de cada lado.

Tiro

O tiro é uma cena independente e sua repetição é possível por meio de um Timer na cena principal. Ao ser disparado, o timer instancia novos tiros e busca os inimigos para direcioná-los. A direção do primeiro projétil é para o inimigo mais próximo, o segundo para o segundo mais próximo e assim em diante até os projéteis daquele disparo terminarem.

Habilidades

As habilidades ficam armazenadas numa classe independente chamada PlayerState. Lá estão propriedades que armazenam informações sobre a velocidade do jogador, velocidade dos projéteis, tempo entre os projéteis e quantidade de projéteis por vez.

O restante do código acessa uma instância dessa classe para alterar os valores, como por exemplo no momento que o jogador escolhe um upgrade. Ao mesmo tempo o código que lida com a velocidade do jogador e com os tiros lê informações da mesma instância.

A tela de upgrade foi implementada como uma cena independente, que é instanciada pela cena principal e exibida por cima dela mesma. Os upgrades

foram modelados como uma enumeração e as probabilidades foram controladas pela repetição de certos valores no array de sorteio.

O upgrade de velocidade do jogador aumenta em 5%, o de tempo entre os tiros é diminuído em 5%, a velocidade do tiro é aumentada em 10% e a quantidade de projéteis por tiro aumenta em 1.

O plano de fundo dessa tela é um ColorRect e o seu conteúdo é um GridContainer com RichTextLabels para os nomes e teclas a serem apertadas.