**GAME DESIGN DOCUMENT:**

**Lala Jump**

Autor: Caio Flávio Almeida Ferreira

Orientador: Roberto de Assis

Conceito

Nome do Jogo: Lala Jump

Estilo do Jogo: Idle game casual

Objetivo do Jogo: O jogador deverá saltar sobre uma barra que estará presa a uma roda, e que a cada segundo irá girar mais rapidamente.

Público Alvo: O jogo pode ser jogado por todos os públicos, mas a faixa etária mais destinada é de 12 a 34 anos.

História: O Jogo não conta com uma história definida por se tratar de um jogo Passa-Tempo com proposta de divertimento rápido.

Principais Regras: O jogo começa quando o jogador tocar na tela para dar início ao jogo. Com o passar do tempo a barra irá girar mais rapidamente e o jogador precisará ser veloz para tocar na tela e pular por cima da barra.

Caso seja acertado pela barra, o jogador perde o jogo e poderá recomeça-lo para uma nova tentativa.

Especificações Técnicas

Hardware: O hardware mínimo testado para a versão final é composto por:

CPU Quad-Core 1.4GHz

Tela: 1280x720

Android Versão 4.3

RAM: 1GB

Qualquer dispositivo que tenha hardware acima do identificado aqui conseguirá executar o jogo.

Sistema Operacional: Inicialmente o jogo estará disponível para o sistema operacional Android v4.3+.

Requerimentos de Software: O único requerimento de software necessário é a versão do Android ser igual ou acima da versão 4.3.

Especificações do Jogo

Número de Fases: O jogo conta com três mapas distintos para serem jogados. Futuras atualizações trarão novos mapas ao jogo

Níveis de Dificuldade: O jogo conta com um único nível de dificuldade. Não será possível a alteração do mesmo.

Modos de Jogo: O jogo conta com doi modos diferentes.

O primeiro single player onde o jogador fará o maior número de pontos possível sem ser acertado pela barra.

O segundo modo é o multiplayer destinado a competição pela internet. O jogador encontrará uma partida online e competirá com outros jogadores para tentar ficar mais tempo possível de pé no jogo. Quando restar um único jogador n competição o jogo pausará e informará ao jogador que ele foi o vencedor da partida.

Em ambos modos o jogador pontuará e essa pontuação será utilizada para aquisição de itens dentro do jogo.

Sistema de pontuação: O jogo conta com dois sistemas de pontuação.

O primeiro é a pontuação local que é a quantidade de saltos que o jogador deu sem se colidir com a barra. Esses saltos serão contabilizados e serão usados como pontos para desbloqueio de itens no inventário.

O segundo tipo de pontuação é o HighScore obtido pelo jogador ao ter pulado o maior número de vezes sobre a barra. Esse HighScore, caso o jogador tenha se registrado, será informado globalmente para todos os demais jogadores. O HighScore será informado caso esteja entre o TOP10 Global, foram isso ele só será visto por seguidores do jogador.

Ranking: O jogo salva o Score mais alto que o jogador fez dentre todas as partidas. Esse ranking é utilizado também para ranquear o jogador globalmente.

Número de Jogadores: O jogo comporta um único jogador por vez.

No modo multiplayer a quantidade de jogadores simultâneos terá capacidade de até 1000 (mil) jogadores.

Arte

Menus: **Figura 1Protagonista**

****

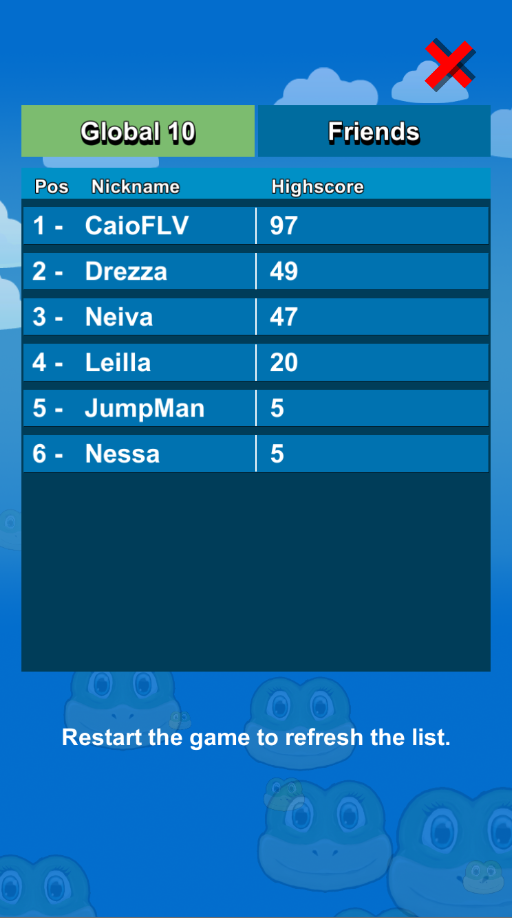
**Figura 2:**  Menu

Menu inicial composto pelos botões que levarão o jogador a cada um dos principais recursos do jogo: Play, Inventário, Loja, Placar de Líderes, Online e configurações.



**Figura 3:** Placar de Líderes

Tela para visualização do TOP10 melhores pontuações globais e aba de amigos onde o jogador pode visualizar o HighScore de amigos.



**Figura 4:** Loja

Área destinada a aquisição de itens Premium(comprados com dinheiro real)



**Figura 5:** Inventário

Área destinada ao inventário de itens do jogador. Por ele, o jogador poderá adicionar ou remover acessórios



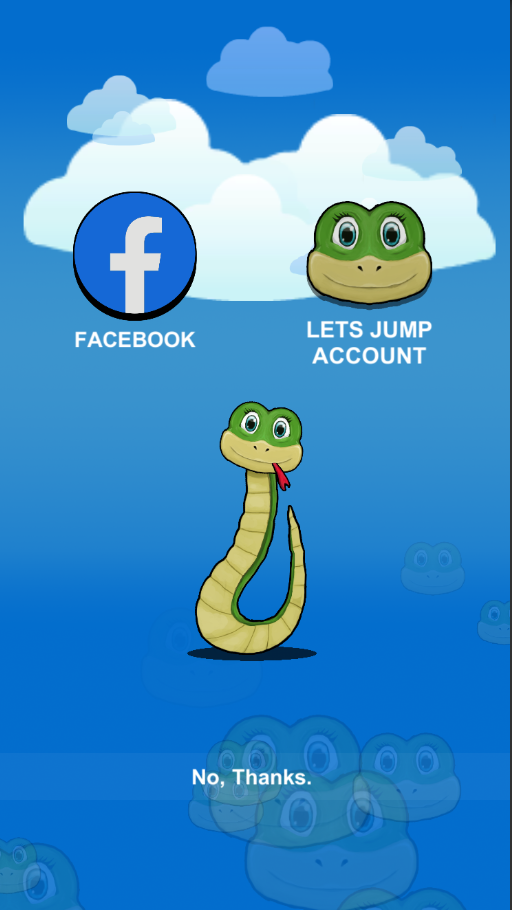
**Figura 6:** Play

Escolha de Mapas para início de partida



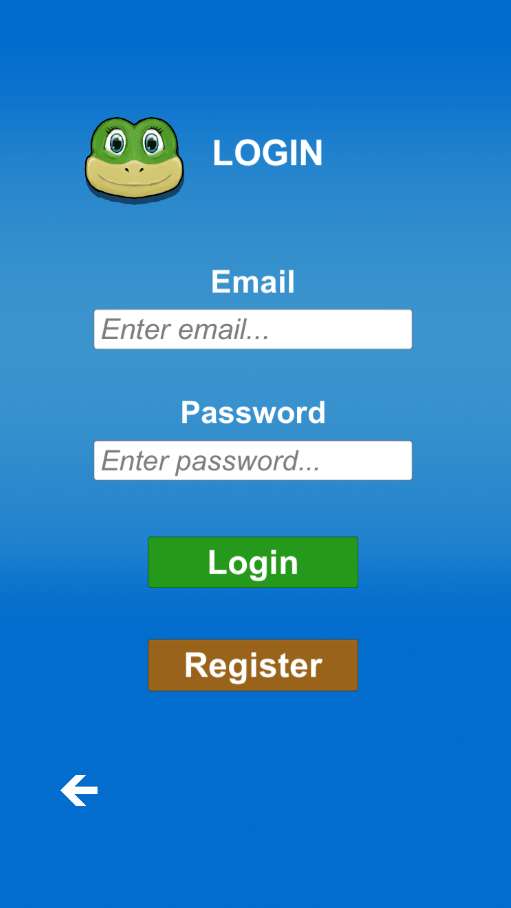
**Figura 7:** Início

Tela inicial do jogo para registro ou login de jogadores



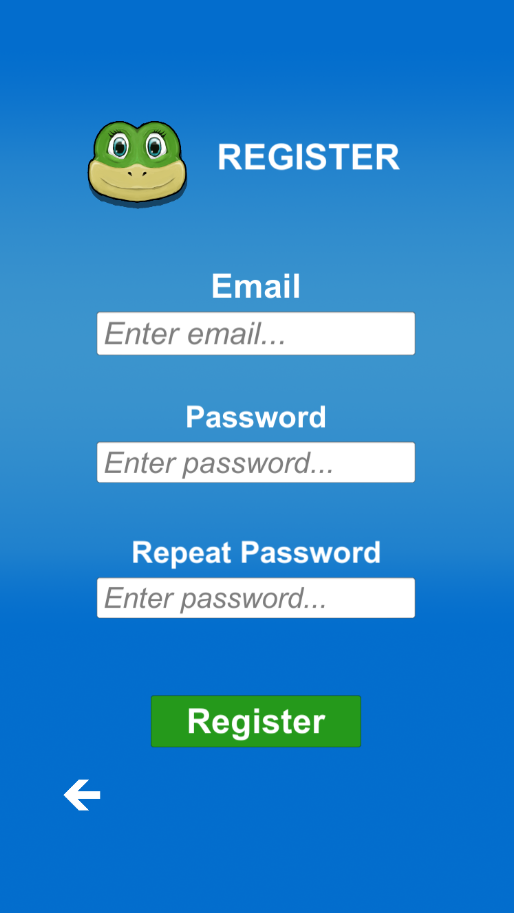
**Figura 8:** Login

Tela de Login de Usuário, informando o e-mail e a senha.

****

**Figura 9:** Registro

Tela de registro de novos usuários



**Figura 10:** Multiplayer

Tela de encontro de partidas multijogador

\

**Figura 11:** Fase 1 – Clássica

Mapa clássico do jogo



**Figura 12:** Fase 2

Mapa 2 do jogo



**Figura 13:** Fase 3

Mapa 3 do jogo



Métricas

Cálculo para Pulo: O cálculo para pulo do personagem principal foi feito para diminuir o impacto de processamento em dispositivos com hardware fraco.

A física envolvendo as bibliotecas de Unity foi totalmente removida dando lugar a cálculos de física básica.

Lançamento Vertical:

transform.localPosition = new Vector2(

transform.localPosition.x,

transform.localPosition.y +

((hmaxPivot.position.y - transform.localPosition.y) \*

jumpVel \* Time.deltaTime) / hmaxPivot.position.y

);

Colisão com a Barra: A colisão com a barra se dá pela distância entre dois pontos, admitindo o Pivot de cada objeto(Centro do jogador e centro da barra).

if ((bar.position - playerCollision.position).magnitude < 2.7f){}

Contador de Pulos: A contagem de pulos é feita a partir de um calculo de ângulos. O sistema analisa a rotação da roda giratória, e se a mesma ultrapassar o ângulo escolhido, um ponto será adicionado ao contador.

Normalmente o ângulo escolhido está situado após a posição do personagem principal, sendo assim, o próprio personagem precisa pular a barra primeiramente para que a roda pontue.

int rot = 145;

if (!lapIsCheck && wheel.eulerAngles.z < rot)

{

lapIsCheck = true;

LapCount++;

lapCountViewer.text = LapCount.ToString();

audioSource.PlayOneShot(lapCompleted, 0.35f);

}

if(wheel.eulerAngles.z > 180)

{

lapIsCheck = false;

}

Sonorização

O jogo conta com dois áudios para pulo do personagem principal.

Cada áudio segue a seguinte regra.

if (UnityEngine.Random.Range(0, 20) < 18)

{

Animator.Play("Jump1", -1, 0);

audioSource.PlayOneShot(Sound1);

}

else

{

Animator.Play("Jump2", -1, 0);

audioSource.PlayOneShot(Sound2, 0.5f);

}

O áudio de pulo é usado de forma randômica.

Multiplayer

Socket.IO: A biblioteca Socket.io é utilizada para conexão com servidor NodeJS hospedado na nuvem.

NodeJS: NodeJS trabalha com a linguagem Javascript para criação de servidores. O servidor criado trabalha da seguinte forma:

NovaConexão > ProcuraPorRoomsLivres > Lobby > BloqueioDaSala > InícioDePartida > FimDePartida > ReciclagemDaSala.

Existem 100(cem) Rooms previamente criadas no servidor, cada room comporta 1000(mil) usuários simultâneos.

Unity + Socket.IO: Na Unity, após a inclusão da biblioteca Socket.io, foram criados vários eventos correspondentes ás funções que se encontram no servidor:

io.On("USER\_QUEUE", OnUserQueue);

io.On("START\_GAME", OnStartGame);

io.On("UPDATE\_PLAYER\_COUNT", OnUpdatePlayerCount);

io.On("USER\_WIN", OnUserWin);

io.On("LEAVE\_GAME", OnLeaveGame);

io.On("END\_GAME", OnEndGame);

io.On("disconnect", OnDisconnect);

Comportamento: Ao clicar em Jogar Online, o jogador envia uma mensagem para o servidor pedindo a entrada em alguma sala disponível. A resposta se dá de duas formas. A primeira significa sucesso na entrada do Lobby, e a segunda significa falha.

Quando uma sala atinge o mínimo para o início de uma partida multijogador, o servidor fecha a sala e envia uma função de início de partida para todos os que se encontram naquela Room.

Casa algum jogador colida com a barra e saia da partida, o mesmo será removido da Room do Servidor. O servidor irá atualizar todos os jogadores com a nova contagem de usuários que restam naquela partida.