Projeto Final TP1 - Flappy Bird

Karen Lima Macêdo [1] Maria Eduarda Machado de Holanda[1]

[1] Universidade de Brasília (UnB), Departamento de Ciência da Computação, Brasil

Figure 1: O projeto consiste na implementação de uma versão do jogo Flappy Bird.

1 Descrição do Problema

A criação de um jogo baseado num jogo já existente chamado Flappy Bird, criado pelo vietnamita Dong Nguyen e que ficou muito famoso em 2013. O objetivo do jogador é bem simples, controlar um pássaro de modo a não deixá-lo colidir com os obstáculos (canos) no caminho. A cada cano que o jogador consegue ultrapassar, ele ganha um ponto. O jogo mostra os scores passados do jogador, assim como seu recorde. O jogo ganhou notoriedade em 2013 pois todos os jogadores queriam bater seus próprios recordes. O cenário do jogo é bem simples, existe um pássaro controlado pelo jogador e surgirá na tela pares de canos com aberturas de tamanhos iguais mas em alturas aleatórias, e o objetivo é ultrapassar esses canos sem colidí-los. Além das funcionalidades existentes do jogo já existente, o jogo também mantém um arquivo com os três melhores recordes e suas respectivas datas, e possibilita ao usuário verificar essas informações sempre que desejado.

2 Definição das Regras de Negócio

As regras são bem simples, o jogador irá mover o pássaro com a barra de espaço de modo a não deixá-lo colidir com os canos no meio do caminho. Apertar a barra de espaço faz com que o pássaro vá para cima, enquanto não apertar faz ele cair. A cada par de canos que o jogador conseguir ultrapassar, soma-se um ponto no seu score. Além disso, quando o jogador ultrapassa um determinado número de canos, a velocidade de passagem do cenário aumenta. O objetivo do jogo é tentar ultrapassar os seus recordes anteriores. Os três primeiros recordes serão armazenados na tela de Score.

3 Classes

3.1 Element

É uma classe abstrata, que serve como superclasse para as classes Bird, Pipe, Score e Button. Sua principal finalidade é fornecer atributos e métodos para desenhar ou atualizar o estado de um objeto na tela.

```
public abstract class Element{
    protected double x;
    protected double y;

    protected int width;
    protected int height;

    public void update(double dt){
    }
    public void draw(Screen s){
    }
}
```

3.2 Bird

Possui a classe Element como herança e uma associação com a classe Box. É a classe responsável por lidar com objeto principal do jogo. Sua associação com a classe box é essencial para detectar possíveis colisões do pássaro com as bordas da tela ou com os canos. No construtor na classe que é definido essa associação, que é atualiza a medida que o pássaro se movimenta. Os métodos `jump()` e `fall()`, são os principais responsáveis por alterar a posição do pássaro na tela, os quais definem sua velocidade de queda ou de subida. Já os métodos sobrescritos atualizam e desenham o pássaro na tela, movimentando-o.

```
public Bird(double x, double y){
    this.x = x;
    this.y = y;
    this.width = 34;
    this.height = 24;
    this.box = new Box(x, y, x + this.width, y +
    this.height);
}
```

3.3 Pipe

Possui a classe Element como herança e uma associação com a classe Box. A classe lida com a criação de um par de canos. Sua associação com a classe Box é primordial para detectar possíveis colisões com o pássaro. É em seu construtor é definida essa associação, sendo atualizada na medida em que os canos se movem na tela. Os pares de canos serão criados na classe principal com alturas aleatórias, mas todos terão um tamanho de abertura fixo: `HOLE_SIZE = 120`, se movimentando na tela com uma `velocity = vx`.

```
public Pipe(double x, double y, double vx){
    this.x = x;
    this.y = y;
    this.width = 52;
    this.height = 270;
    this.velocity = vx;
```

```
    boxTop = new Box(x, y - this.height, x + this.
    width, y);
    boxBottom = new Box(x, y+ Pipe.HOLE_SIZE, x+
    this.width, y+ Pipe.HOLE_SIZE + this.height);
}

@Override
public void update(double dt) {
    this.move(dt);
    boxTop.move(velocity * dt, 0);
    boxBottom.move(velocity * dt, 0);
}
```

3.4 Box

Essa classe é responsável por detectar possíveis colisões do pássaro com os canos. Ela atualiza constantemente a posição desses objetos na tela. Os atributos `x0, x1, y0, y1` desenham uma espécie de caixa, que contorna o tamanho do objeto, com isso é possível determinar se houve ou não colisão.

```
public boolean collision(Box box) {
    // Falso se nao tiver colisao
    if (this.x1 < box.x0 || this.x0 > box.x1)
        return false;
    else if (this.y1 < box.y0 || this.y0 > box.y1)
        return false;

    // Verdadeiro se tiver colisao
    return true;
}
```

3.5 Button

Possui a classe Element como herença e é responsável por criar um botão na tela. Existem 3 tipos de botões que serão usados: `Start, Scores, Menu`. Ao clicar em cada um deles é possível alterar a tela e o estado do jogo. A classe é responsável por desenhar cada botão e determinar se foi clicado ou não em algum botão em questão no método abaixo:

```
// checa se o botao foi pressionado ou nao
public boolean pointOnButton(double x, double y){
    return (x) > this.x && x < (this.x + this.
    width) &&
           (y) > this.y && y < (this.y + this.
    height);
}
```

3.6 Score

Possui a classe Element como herança e é responsável por desenhar tanto o score atual do jogador, como os seus recordes, assim como atualizá-los. Nessa classe temos o score como inteiro e como string, para ficar mais fácil de manipular o sprite de imagens.

3.7 Animation

Responsável por trabalhar com a interface gráfica, criar as telas e suas configurações. Também mantém a tela rodando e sua animação até que alguma ação do usuário acione um evento. Além disso, inicia e encerra o jogo. Possui uma associação com a interface Game. É no seu construtor que é feito todo o tratamento da interface gráfica, do teclado e do mouse.

```
// loop principal
private void mainLoop() {
    while (this.running) {
        if (!this.running) {
            return;
        }
        this.t1 = System.currentTimeMillis();
        this.dt = (this.t1 - this.t0) * 0.001;
        this.t0 = this.t1;

        game.step(this.dt);
        Graphics2D g = (Graphics2D) strategy.
    getDrawGraphics();
        game.draw(new Screen(g));
        strategy.show();
    }
}
```

3.8 Screen

Classe responsável por desenhar todas as imagens na tela, de dois jeitos possíveis, com a imagem padrão definida pela classe, ou com um arquivo passado como parâmetro em formato de String.

```
Graphics2D g;
public ImageIcon img= new ImageIcon("flappy.png");

public void drawImage(int xa, int ya, int larg,
    int alt, double dir, double x, double y) {
    /* implementacao */
}

public void drawImage(String img, int xa, int ya,
    int larg, int alt, double dir, double x,
    double y) {
    /* implementacao */
}
```

3.9 Game

Interface para a classe principal FlappyBird, possui os métodos principais que serão implementados.

```
public interface Game {

    public static int WIDTH = 384;
    public static int HEIGHT = 512;
    public static String TITLE = "Flappy Bird";

    boolean isGameOver();

    void step(double dt);

    void draw(Screen tela);

    void onKeyPressed(String tecla);

    void onMouseClicked(double x, double y);

    void onMouseMoved(Canvas canvas, double x,
    double y);
}
```

3.10 FlappyBird

Essa é a classe principal do jogo, nela são implementados todos os métodos para execução do jogo. Ela implementa a interface Game. Nela existem 4 estados do jogo:

```
public enum STATE {
    MENU, READY, START, RECORDS;
}
```

Esses estados são alterados apenas clicando em cada botão. O jogo inicia no estado `MENU`, e poderá ser alterado para o estado `RECORDS` se clicarmos no botão Score ou para o estado `READY` se

clicarmos no botão Start, e mudará automaticamente após 2 segundos para o estado `START`, onde começará o jogo em si. Se o pássaro colidir com as extremidades da tela ou com os canos, acabará o jogo e você terá a opção de voltar para a tela de Menu.

A lógica inteira do jogo está implementada aqui. Existe, por exemplo, o método `addPipes()` que randomiza as alturas que os canos irão aparecer na tela. Mas os dois métodos principais são: `step()`, que atualiza o jogo a cada frame, e `draw()`, que desenhará cada tela possível dependendo do estado que se encontra o jogo. Além disso, é nessa classe que são manipulados os arquivos para salvar os recordes do jogador.

3.11 Files

É responsável por acessar os arquivos que guardam na memória os melhores recordes e suas respectivas datas. A classe recupera esses dados antes do início do jogo e grava novamente fazendo as alterações necessárias após o fim do jogo. Para isso, possui um método read para cada arquivo assim como um método write.

3.12 Diagrama de Classes

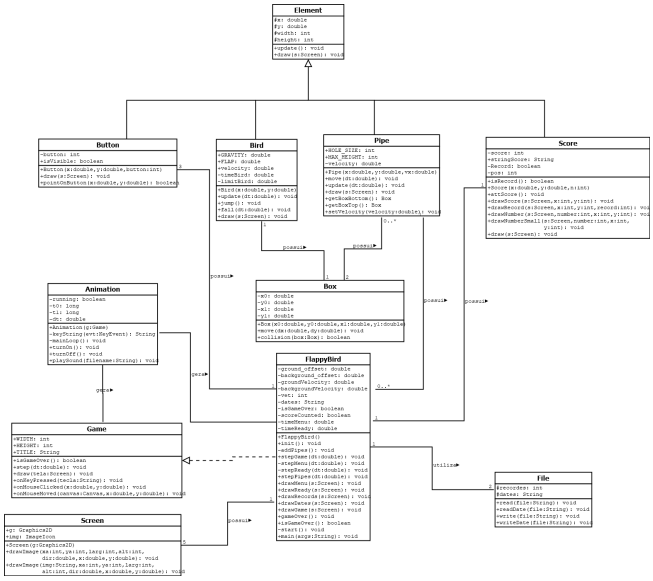

Figure 2: Diagrama de Classes

4 TELAS

4.1 Tela Inicial

É a primeira tela aberta quando o usuário clica no executável. Apresenta o menu com duas áreas clicáveis: START e SCORE, que desencadeiam novos eventos. Clicando em START uma nova tela é desenhada avisando pro usuário se preparar pro início do jogo. E clicando em SCORE é possível visualizar os três melhores recordes e suas respectivas datas. Essa tela é animada, o cenário e o chão se movimentam da direita para esquerda e o pássaro permanece na mesma posição batendo as asas.

4.2 Tela de Recordes

Apresenta uma área clicável: MENU, que permite ao usuário retornar a tela inicial do jogo. Mostra os três melhores recordes em ordem decrescente e suas respectivas datas. Também é uma tela animada, o chão e o cenário se movimentam da direita para esquerda.

4.3 Tela que antecede o início do jogo

Tela animada com o chão e o cenário se movimentando da direita para esquerda por alguns segundos. Apresenta um título de aviso para o usuário se preparar para o início do jogo.

4.4 Tela do Jogo

É a tela principal do projeto. O cenário, o chão e os canos se movimentam da direita para a esquerda. O pássaro permanece na mesma posição horizontalmente mas muda de posição verticalmente, ele está em constante queda enquanto o usuário não aperta o teclado. Também possui um movimento de rotação, quando está em queda a rotação acontece no sentido horário e na subida a rotação acontece no sentido anti-horário. A abertura entre os canos superiores e inferiores é sempre a mesma porém a posição vertical muda aleatoriamente. O espaçamento horizontal entre os canos também é sempre o mesmo. No canto superior esquerdo é mostrado o score do usuário de acordo com o avanço dele no jogo. Essa tela permanece desenhada até o momento que o pássaro atinge o chão ou algum dos canos.

4.5 Tela de Game Over

É desenhada quando o pássaro colide com o chão ou um cano. Também é uma tela animada, o cenário e o chão se movimentam da direita para a esquerda. Apresenta um quadro centralizado que informa para o usuário o score obtido e o melhor recorde, caso o score dele seja melhor, também é desenhado na tela uma medalha de ouro. É desenhado uma medalha de prata ou de bronze caso o score seja melhor que o segundo ou o terceiro melhores recordes. Possui uma área clicável que permite ao usuário retornar para tela inicial do jogo.

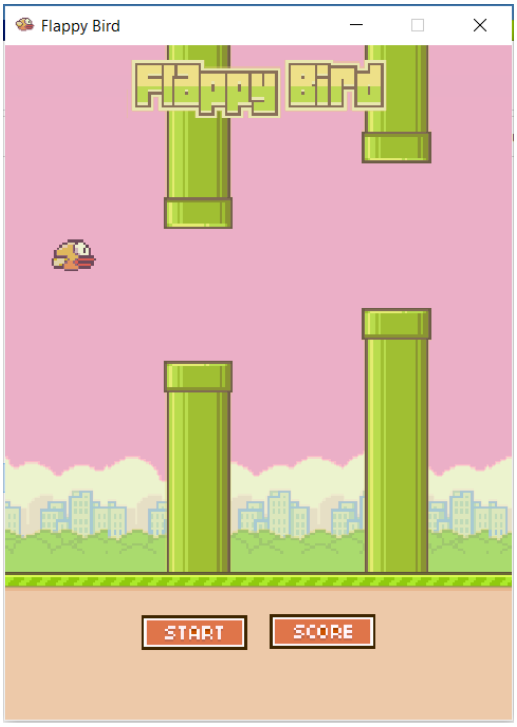

Figure 3: Tela Inicial

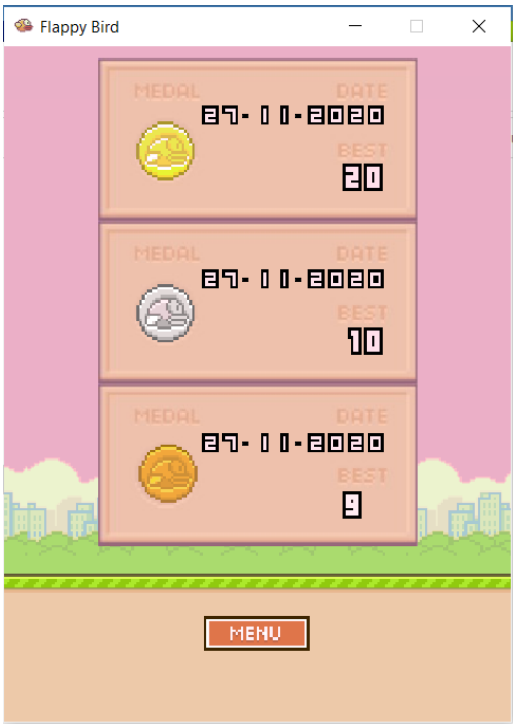

Figure 4: Tela Recordes

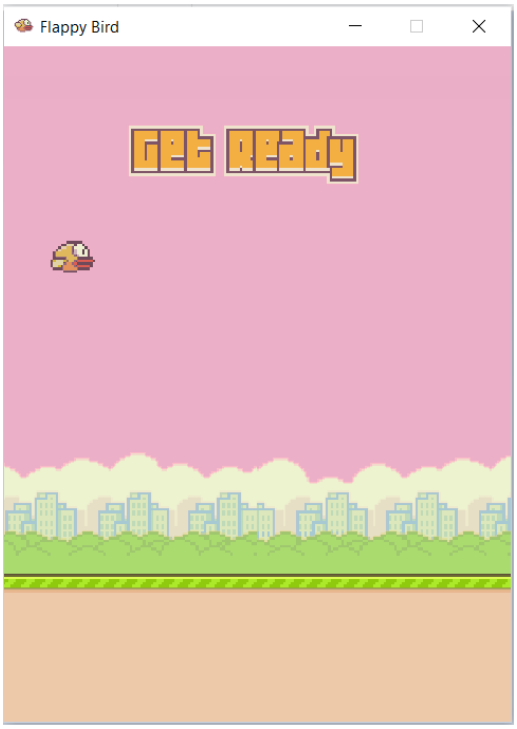

Figure 5: Tela Get Ready

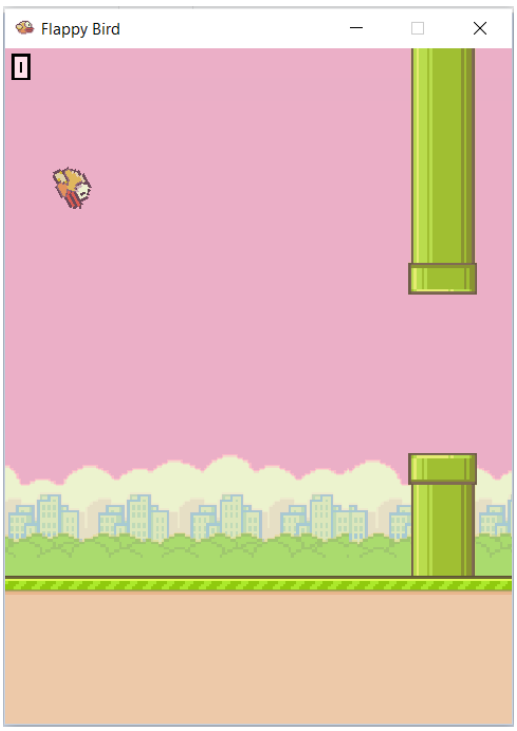

Figure 6: Tela Jogo

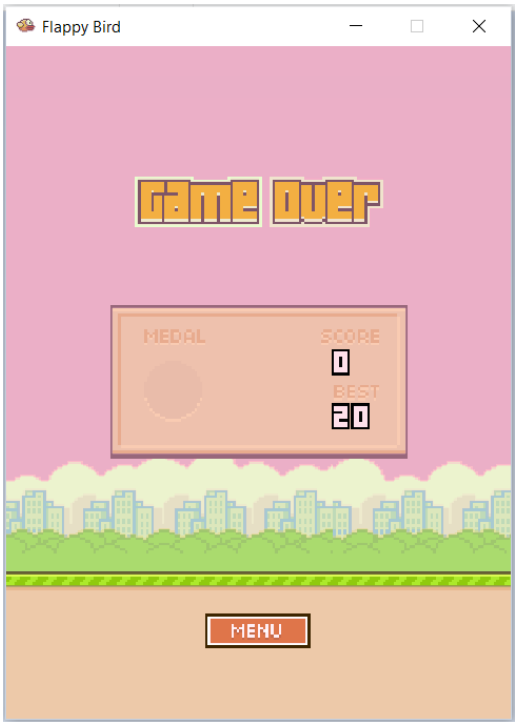

Figure 7: Tela Game Over