libGDX

https://github.com/libgdx/libgdx/wiki/

Rafael Vieira Coelho

rafael.coelho@farroupilha.ifrs.edu.br

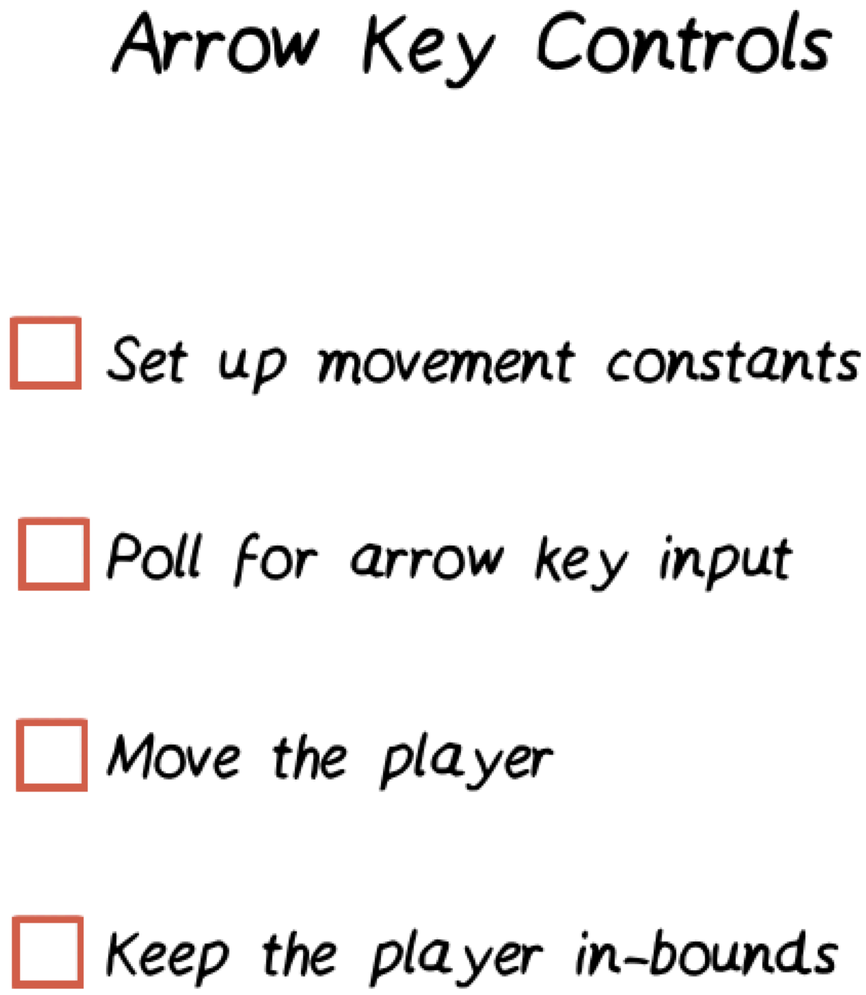
Parte 6 - Jogo Completo

- Versão 1 - Construção do Projeto
- Versão 2 - Desenha icicle
- Versão 3 - Desenha Jogador
- **Versão 4 - Controle de Teclado (setas)**
- Versão 5 - Adiciona Icicles
- Versão 6 - Remove Icicles que somem da tela
- Versão 7 - Detecta Colisão
- Versão 8 - Adiciona o HUD
- Versão 9 - Adiciona níveis de dificuldade
- Versão 10 - Adiciona seleção de nível de dificuldade

https://github.com/libgdx/libgdx/wiki/

Como Movimento o Jogador?

1. Definimos as constantes de movimento
2. Obtemos a entrada do teclado
3. Movemos o jogador
4. Controlamos o limite da tela para o jogador não sair



Classe Constants.java

- Adicionamos a constante que define a velocidade do jogador ao se movimentar

Constants.java

```
1   package com.udacity.gamedev.icicles;
2
3   import com.badlogic.gdx.graphics.Color;
4
5
6   public class Constants {
7       public static final float WORLD_SIZE = 10.0f;
8       public static final Color BACKGROUND_COLOR = Color.BLUE;
9
10      public static final float PLAYER_HEAD_RADIUS = 0.5f;
11      public static final float PLAYER_HEAD_HEIGHT = 4.0f * PLAYER_HEAD_RADIUS;
12      public static final float PLAYER_LIMB_WIDTH = 0.1f;
13      public static final int PLAYER_HEAD_SEGMENTS = 20;
14      public static final Color PLAYER_COLOR = Color.BLACK;
15
16      // TODO: Add Constant for player movement speed
17      public static final float PLAYER_MOVEMENT_SPEED = 10.0f;
18
19      public static final float ICICLES_HEIGHT = 1.0f;
20      public static final float ICICLES_WIDTH = 0.5f;
21      public static final Color ICICLE_COLOR = Color.WHITE;
22
23  }
```

Classe Player.java

- Testamos a entrada do teclado através do módulo Gdx.input com o método isKeyPressed() que verifica se a tecla passada como parâmetro foi pressionada (constantes em Keys.java)

Player.java

```
27
28      public void update(float delta) {
29          // TODO: Use Gdx.input.isKeyPressed() to move the player in
30          //  the appropriate direction when an arrow key is pressed
31          if (Gdx.input.isKeyPressed(Keys.LEFT)) {
32              position.x -= delta * Constants.PLAYER_MOVEMENT_SPEED;
33          } else if (Gdx.input.isKeyPressed(Keys.RIGHT)) {
34              position.x += delta * Constants.PLAYER_MOVEMENT_SPEED;
35          }
36
37          ensureInBounds();
38      }
```

Classe Player.java

- E precisamos limitar a movimentação do jogador caso ele exceda os limites do nosso jogo (inferior a zero ou superior a largura da tela - getWorldWidth())

Player.java

```
39
40      private void ensureInBounds() {
41          // TODO: Complete this function to ensure the player is within the viewport
42          if (position.x - Constants.PLAYER_HEAD_RADIUS < 0) {
43              position.x = Constants.PLAYER_HEAD_RADIUS;
44          }
45          if (position.x + Constants.PLAYER_HEAD_RADIUS > viewport.getWorldWidth()) {
46              position.x = viewport.getWorldWidth() - Constants.PLAYER_HEAD_RADIUS;
47          }
48      }
```

Player.java

E por fim, desenhar novamente o jogador.

```
50  @   public void render(ShapeRenderer renderer) {
51          renderer.setColor(Constants.PLAYER_COLOR);
52          renderer.set(ShapeType.Filled);
53          renderer.circle(position.x, position.y, Constants.PLAYER_HEAD_RADIUS, Constants.PLAYER_HEAD_SEGMENTS);
54
55          Vector2 torsoTop = new Vector2(position.x,  y: position.y - Constants.PLAYER_HEAD_RADIUS);
56          Vector2 torsoBottom = new Vector2(torsoTop.x,  y: torsoTop.y - 2 * Constants.PLAYER_HEAD_RADIUS);
57
58          renderer.rectLine(torsoTop, torsoBottom, Constants.PLAYER_LIMB_WIDTH);
59
60          renderer.rectLine(
61                  torsoTop.x, torsoTop.y,
62                  x2: torsoTop.x + Constants.PLAYER_HEAD_RADIUS,  y2: torsoTop.y - Constants.PLAYER_HEAD_RADIUS,
63                  Constants.PLAYER_LIMB_WIDTH);
64          renderer.rectLine(
65                  torsoTop.x, torsoTop.y,
66                  x2: torsoTop.x - Constants.PLAYER_HEAD_RADIUS,  y2: torsoTop.y - Constants.PLAYER_HEAD_RADIUS,
67                  Constants.PLAYER_LIMB_WIDTH);
68
69          renderer.rectLine(
70                  torsoBottom.x, torsoBottom.y,
71                  x2: torsoBottom.x + Constants.PLAYER_HEAD_RADIUS,  y2: torsoBottom.y - Constants.PLAYER_HEAD_RADIUS,
72                  Constants.PLAYER_LIMB_WIDTH);
73
74          renderer.rectLine(
75                  torsoBottom.x, torsoBottom.y,
76                  x2: torsoBottom.x - Constants.PLAYER_HEAD_RADIUS,  y2: torsoBottom.y - Constants.PLAYER_HEAD_RADIUS,
77                  Constants.PLAYER_LIMB_WIDTH);
78      }
```