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 Remover Blocos de Gelo que não Aparecem mais?

1. Mover os icicles que saíram da tela para o DelayedRemovalArray

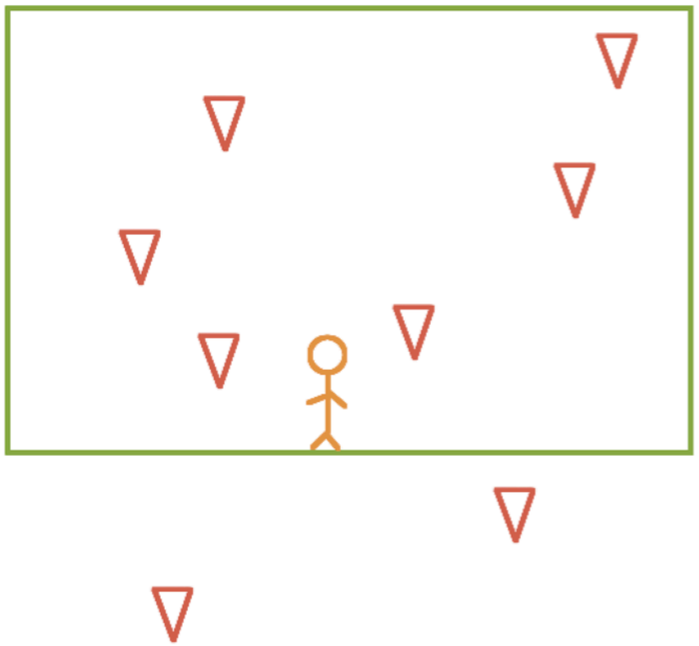
2. Iniciar sessão de remoção

3. Remover os icicles que saíram da tela

4. Finalizar sessão de remoção

Remove Stale Icicles

- [ ] Switch to DelayedRemovalArray
- [ ] Begin removal session
- [ ] Remove fallen icicles
- [ ] End removal session



Classe DelayedRemovalArray.java

- Trata-se de uma classe do próprio LibGDX que é usado para remoção de objetos da tela usando delay para não ocorrer abruptamente.

Icicles.java | DelayedRemovalArray.java

```
22      * {@link #begin()} is called to occur once {@link #end()} is called. This can
23      * without affecting iteration. Between begin and end, most mutator methods wi
24      * {@link #removeIndex(int)}, {@link #removeValue(Object, boolean)}, {@link #r
25      * methods are allowed.
26      * <p>
27      * Note that DelayedRemovalArray is not for thread safety, only for removal du
28      * <p>
29      * Code using this class must not rely on items being removed immediately. Con
30      * problem.
31      * @author Nathan Sweet */
32     public class DelayedRemovalArray<T> extends Array<T> {
33         private int iterating;
34         private IntArray remove = new IntArray( capacity: 0);
35         private int clear;
36
37         public DelayedRemovalArray () { super(); }
40
41 @       public DelayedRemovalArray (Array array) { super(array); }
44
45         public DelayedRemovalArray (boolean ordered, int capacity, Class arrayType
48
49         public DelayedRemovalArray (boolean ordered, int capacity) { super(ordered
52
53 @       public DelayedRemovalArray (boolean ordered, T[] array, int startIndex, in
54             super(ordered, array, startIndex, count);
55         }
56
57         public DelayedRemovalArray (Class arrayType) { super(arrayType); }
```

Classe Icicles.java

- Criamos o ArrayList especial para armazenar os icicles que devem ser removidos.

- E inicializarmos no método **init()**

Icicles.java

```
1   package com.udacity.gamedev.icicles;
2
3   import com.badlogic.gdx.graphics.glutils.ShapeRenderer;
4   import com.badlogic.gdx.math.MathUtils;
5   import com.badlogic.gdx.math.Vector2;
6   import com.badlogic.gdx.utils.DelayedRemovalArray;
7   import com.badlogic.gdx.utils.viewport.Viewport;
8
9   public class Icicles {
10
11      public static final String TAG = Icicles.class.getName();
12
13      // TODO: Use a DelayedRemovalArray to hold our icicles
14      DelayedRemovalArray<Icicle> icicleList;
15      Viewport viewport;
16
17      public Icicles(Viewport viewport) {
18          this.viewport = viewport;
19          init();
20      }
21
22      public void init() {
23          // TODO: Initialize the DelayedRemovalArray
24          icicleList = new DelayedRemovalArray<Icicle>( ordered: false,  capacity: 100);
25      }
```

Classe Icicles.java

- Iniciamos a sessão com o método **begin()**
- Devemos percorrer o **DelayedRemovedArray** e testar se a posição y extrapolou a tela
- Finalizamos a sessão com o método **end()**

Icicles.java

```
27      public void update(float delta) {
28          if (MathUtils.random() < delta * Constants.ICICLE_SPAWNS_PER_SECOND) {
29              Vector2 newIciclePosition = new Vector2(
30                      x: MathUtils.random() * viewport.getWorldWidth(),
31                      viewport.getWorldHeight()
32              );
33              Icicle newIcicle = new Icicle(newIciclePosition);
34              icicleList.add(newIcicle);
35          }
36
37          for (Icicle icicle : icicleList) {
38              icicle.update(delta);
39          }
40
41          // TODO: begin a removal session
42          icicleList.begin();
43
44          // TODO: Remove any icicle completely off the bottom of the screen
45          for (int i = 0; i < icicleList.size; i++) {
46              if (icicleList.get(i).position.y < -Constants.ICICLES_HEIGHT) {
47                  icicleList.removeIndex(i);
48              }
49          }
50          // TODO: End removal session
51          icicleList.end();
52      }
```