package com.mygdx.game;

import com.badlogic.gdx.ApplicationAdapter;

import com.badlogic.gdx.Gdx;

import com.badlogic.gdx.graphics.Color;

import com.badlogic.gdx.graphics.Texture;

import com.badlogic.gdx.graphics.g2d.SpriteBatch;

import com.badlogic.gdx.math.Intersector;

import com.badlogic.gdx.math.Rectangle;

import com.badlogic.gdx.graphics.g2d.BitmapFont;

import java.util.Random;

public class Googledino extends ApplicationAdapter {

SpriteBatch batch;

Texture background,gameOver, land, cloud,restart;

Texture [] dino;

Texture [] cactii;

int dinoState =1, gameState=0;

float dinoY;

int n;

int[] cactusX = new int[4];

int cactusDist;

BitmapFont font;

Rectangle[] cactiiRectangle;

Rectangle dinoRectangle;

int score = 0; int scoringCactus = 0;

@Override

public void create () {

batch = new SpriteBatch();

Random rand = new Random();

background = new Texture("white.png");

gameOver = new Texture("gameover.png");

land= new Texture("land.png");

cloud= new Texture("cloud.png");

restart= new Texture("restart.png");

// birdCircle = new Circle();

font = new BitmapFont();

font.setColor(Color.BLACK);

font.getData().setScale(7);

dino= new Texture[3];

dino[0] = new Texture("dino2.png");

dino[1] = new Texture("dino3.png");

dino[2] = new Texture("dino.png");

dinoY=Gdx.graphics.getHeight()/2;

cactusDist = rand.nextInt(Gdx.graphics.getWidth()\*2);

// random = new Random();

cactii= new Texture[4];

cactii[0]= new Texture("cactus.png");

cactii[1]= new Texture("cactus2.png");

cactii[2]= new Texture ("cactus3.png");

cactii[3]= new Texture("cactus4.png");

cactiiRectangle=new Rectangle[4];

for(int i=0; i<4;i++)

{n=rand.nextInt(4);

cactusX[i] = Gdx.graphics.getWidth()/2-cactii[n].getWidth()/2 + i\*cactusDist + Gdx.graphics.getWidth() ;

cactiiRectangle[i]= new Rectangle();

}

dinoRectangle=new Rectangle();

}

@Override

public void render () {

batch.begin();

batch.draw(background, 0, 0, Gdx.graphics.getWidth(), Gdx.graphics.getHeight());

batch.draw(land, 0, Gdx.graphics.getHeight() / 2);

batch.draw(cloud, Gdx.graphics.getWidth() \* 3 / 4, Gdx.graphics.getHeight() \* 3 / 4);

batch.draw(cloud, Gdx.graphics.getWidth() \* 5 / 6, Gdx.graphics.getHeight() \* 5 / 6);

if (Gdx.input.justTouched()) {

if (gameState == 2) {

for (int i = 0; i < 4; i++) {

cactusX[i] = Gdx.graphics.getWidth() / 2 - cactii[n].getWidth() / 2 + i \* cactusDist + Gdx.graphics.getWidth();

cactiiRectangle[i] = new Rectangle();

}

}

dinoY = Gdx.graphics.getHeight() / 2 + 150;

gameState = 1;

}

if (gameState == 1) {

if (dinoY > Gdx.graphics.getHeight() / 2) {

dinoY = dinoY - 5;

}

if (dinoState == 0) {

dinoState = 1;

} else {

dinoState = 0;

}

for (int i = 0; i < 4; i++) {

if (cactusX[i] < -cactii[i].getWidth()) {

cactusX[i] += 4 \* cactusDist;

} else {

cactusX[i] = cactusX[i] - 10;

}

if (cactusX[scoringCactus] < Gdx.graphics.getWidth() / 2) {

score++;

if (scoringCactus < 3) {

scoringCactus++;

} else

scoringCactus = 0;

}

batch.draw(cactii[n], cactusX[i], Gdx.graphics.getHeight() / 2);

cactiiRectangle[i] = new Rectangle().set(cactusX[i], (Gdx.graphics.getHeight() / 2), cactii[n].getWidth(), cactii[n].getHeight());

if (Intersector.overlaps(dinoRectangle, cactiiRectangle[i])) {

gameState = 2;

}

font.draw(batch, String.valueOf(score), 200, 1500);

}

} else if (gameState == 2) {

for (int i = 0; i < 4; i++) {

batch.draw(cactii[n], cactusX[i], Gdx.graphics.getHeight() / 2);

}

scoringCactus = 0;

score = 0;

batch.draw(gameOver, Gdx.graphics.getWidth() / 2 - gameOver.getWidth() / 2, Gdx.graphics.getHeight() / 2 + 500);

batch.draw(restart,Gdx.graphics.getWidth()/2-restart.getWidth()/2,Gdx.graphics.getHeight()/2+200);

if (Gdx.input.justTouched()) {

gameState = 1;

}

font.draw(batch, String.valueOf(score), 200, 1500);

}

batch.draw(dino[dinoState], Gdx.graphics.getWidth() / 3, dinoY);

dinoRectangle.set(Gdx.graphics.getWidth() / 3, dinoY, dino[dinoState].getWidth(), dino[dinoState].getHeight());

batch.end();

}

@Override

public void dispose() {

batch.dispose();

background.dispose();

}

}