DESERT GAME

package com.game.desert;  
  
import com.badlogic.gdx.Game;  
import com.badlogic.gdx.assets.AssetManager;  
import com.badlogic.gdx.audio.Music;  
import com.badlogic.gdx.audio.Sound;  
import com.badlogic.gdx.graphics.g2d.SpriteBatch;  
import com.game.desert.Screens.\*;  
  
public class DesertGame extends Game {  
 //Ekranların nesnelerini oluşturup onlara bir sıra atıyoruz.  
 private LoadingScreen loadingScreen;  
 private PreferencesScreen preferencesScreen;  
 private MenuScreen menuScreen;  
 private PlayScreen playScreen;  
 private PauseScreen pauseScreen;  
 private AppPreferences preferences;  
 //Müzik ve oyun skalası için gerekli değerler  
 private Music music;  
 public static AssetManager *manager*;  
 //Ekran sıraları  
 public final static int *MENU* = 0;  
 public final static int *PREFERENCES* = 1;  
 public final static int *APPLICATION* = 2;  
 public static final int *PAUSE* = 3;  
 public static final int *WIDTH* = 1280;  
 public static final int *HEIGHT* = 720;  
 public static final float *PPM* = 100;  
 public SpriteBatch batch;  
  
  
  
 @Override  
 public void create() {  
 //Oyun açılırken gelen ekran  
 batch = new SpriteBatch();  
 loadingScreen = new LoadingScreen(this);  
 preferences = new AppPreferences();  
  
 *manager* = new AssetManager();  
 *manager*.load("Music/arkaplan\_1.ogg", Music.class);  
 *manager*.load("Music/yurume\_sesi\_1.ogg", Sound.class);  
 *manager*.finishLoading();  
  
 setScreen(loadingScreen);  
 }  
 public void changeScreen(int screen){  
 //Ekran değişimlerini sağlayan method  
 switch(screen){  
 case *MENU*:  
 if(menuScreen == null) menuScreen = new MenuScreen(this);  
 this.setScreen(menuScreen);  
 break;  
 case *PREFERENCES*:  
 if(preferencesScreen == null) preferencesScreen = new PreferencesScreen(this);  
 this.setScreen(preferencesScreen);  
 break;  
 case *APPLICATION*:  
 if(playScreen == null) playScreen = new PlayScreen(this);  
 this.setScreen(playScreen);  
 break;  
 case *PAUSE*:  
 if(pauseScreen == null) pauseScreen = new PauseScreen(this);  
 this.setScreen(pauseScreen);  
 break;  
 }  
 }  
 public AppPreferences getPreferences(){  
 //Ayarların kaydedilmesini sağlar.  
 return this.preferences;  
 }  
  
 @Override  
 public void render(){  
 //Müziğin çalınması için gerekli ayarlar.  
 if(getPreferences().isMusicEnabled()){  
 music = DesertGame.*manager*.get("Music/arkaplan\_1.ogg", Music.class);  
 music.setLooping(true);  
 music.setVolume(music.getVolume());  
 music.play();  
 }  
 else{  
 music = DesertGame.*manager*.get("Music/arkaplan\_1.ogg", Music.class);  
 music.setLooping(true);  
 music.setVolume(music.getVolume());  
 music.stop();  
 }  
 super.render();  
 *manager*.update();  
 }  
  
 @Override  
 public void dispose(){  
 batch.dispose();  
 }  
  
  
}

PLAYSCREEN

package com.game.desert.Screens;  
  
  
import com.badlogic.gdx.Gdx;  
import com.badlogic.gdx.Input;  
import com.badlogic.gdx.Screen;  
import com.badlogic.gdx.audio.Music;  
import com.badlogic.gdx.audio.Sound;  
import com.badlogic.gdx.graphics.GL20;  
import com.badlogic.gdx.graphics.OrthographicCamera;  
import com.badlogic.gdx.graphics.g2d.TextureAtlas;  
import com.badlogic.gdx.maps.MapObject;  
import com.badlogic.gdx.maps.objects.RectangleMapObject;  
import com.badlogic.gdx.maps.tiled.TiledMap;  
import com.badlogic.gdx.maps.tiled.TmxMapLoader;  
import com.badlogic.gdx.maps.tiled.renderers.OrthogonalTiledMapRenderer;  
import com.badlogic.gdx.math.Rectangle;  
import com.badlogic.gdx.math.Vector2;  
import com.badlogic.gdx.physics.box2d.\*;  
import com.badlogic.gdx.utils.viewport.FitViewport;  
import com.badlogic.gdx.utils.viewport.Viewport;  
import com.game.desert.Sprites.Character;  
import com.game.desert.DesertGame;  
  
public class PlayScreen implements Screen {  
  
 private TextureAtlas atlas; // Karakter animasyonu için eklenen nesne  
 private DesertGame game;  
  
 private Body b2body;  
 //Camera için kullanılan sınıf  
 private OrthographicCamera gameCam;  
 private Viewport gamePort;  
 //Tiled uygulamasıyla haritayı koda aktarır  
 private TmxMapLoader mapLoader;  
 private TiledMap map;  
 private OrthogonalTiledMapRenderer renderer;  
 //Oyun dünyası için değişkenler  
 private World world;  
 private Character player;  
 private int jumpCounter = 0;  
 //Ses efektleri  
 private Music music;  
 private Sound sound;  
 private boolean checkPoint= false;  
  
 Music music1 = Gdx.*audio*.newMusic(Gdx.*files*.internal("yurume\_sesi\_1\_1.ogg"));  
 Music music2 = Gdx.*audio*.newMusic(Gdx.*files*.internal("ziplama\_sesi\_2.ogg"));  
  
 public PlayScreen(DesertGame game){  
  
 atlas= new TextureAtlas("Karakter.pack"); //Karakter animasyonlarını içeren dosya  
  
 this.game = game;  
  
 //Kameranın oluşturulması  
 gameCam = new OrthographicCamera();  
 gamePort = new FitViewport(DesertGame.*WIDTH*/DesertGame.*PPM*, DesertGame.*HEIGHT*/DesertGame.*PPM*, gameCam);  
  
 //Haritanın tanımlanması  
 mapLoader = new TmxMapLoader();  
 map = mapLoader.load("adsız.tmx");  
 renderer = new OrthogonalTiledMapRenderer(map, 1/DesertGame.*PPM*);  
 gameCam.position.set(gamePort.getWorldWidth()/2, gamePort.getWorldHeight()/2, 0);  
  
 //Oyun dünyasının oluşturulması  
 world = new World(new Vector2(0,-10), true);  
 player = new Character(world,this);  
  
  
  
 //Vücudun oluşturulması  
 BodyDef bodyDef = new BodyDef();  
 PolygonShape polygonShape = new PolygonShape();  
 FixtureDef fixtureDef = new FixtureDef();  
 Body body;  
  
 //Oyundaki zeminin oluşturulması  
 for(MapObject object : map.getLayers().get(2).getObjects().getByType(RectangleMapObject.class)){  
 Rectangle rectangle =((RectangleMapObject) object).getRectangle();  
  
 bodyDef.type = BodyDef.BodyType.*StaticBody*;  
 bodyDef.position.set((rectangle.getX() + rectangle.getWidth()/2)/DesertGame.*PPM*, (rectangle.getY() + rectangle.getHeight()/2)/DesertGame.*PPM*);  
 body = world.createBody(bodyDef);  
 polygonShape.setAsBox(rectangle.getWidth()/2/DesertGame.*PPM*, rectangle.getHeight()/2/DesertGame.*PPM*);  
 fixtureDef.shape = polygonShape;  
 body.createFixture(fixtureDef);  
 }  
 }  
  
  
  
 public TextureAtlas getAtlas(){  
  
 return atlas;  
 }  
  
 @Override  
 public void show() {  
  
 }  
  
 public void handleInput(float delta) {  
  
 //Karakter kontrolü ve ses efektleri için gerekli kodlar  
 if (Gdx.*input*.isKeyPressed(Input.Keys.*UP*) && jumpCounter < 1) {  
 player.b2body.applyLinearImpulse(new Vector2(0, 4f), player.b2body.getWorldCenter(), true);  
 jumpCounter++;  
 music2.play();  
 }  
 if (Gdx.*input*.isKeyPressed(Input.Keys.*RIGHT*) && player.b2body.getLinearVelocity().x <= 3) {  
 player.b2body.applyLinearImpulse(new Vector2(0.075f, 0), player.b2body.getWorldCenter(), true);  
 music1.play();  
 }  
  
 if (Gdx.*input*.isKeyPressed(Input.Keys.*LEFT*) && player.b2body.getLinearVelocity().x >= -3) {  
 player.b2body.applyLinearImpulse(new Vector2(-0.075f, 0), player.b2body.getWorldCenter(), true);  
 music1.play();  
 }  
  
 if (player.b2body.getLinearVelocity().y == 0) {  
 jumpCounter = 0;  
 }  
  
 if(Gdx.*input*.isKeyPressed(Input.Keys.*ESCAPE*)){  
 game.changeScreen(DesertGame.*PAUSE*);  
 }  
 }  
  
  
 private void update(float delta) {  
  
 //Oyunun yenilenmesi için gerekli kodlar  
 handleInput(delta);  
 world.step(1/60f, 6, 2);  
 gameCam.position.x = player.b2body.getPosition().x;  
 gameCam.update();  
 renderer.setView(gameCam);  
 player.update(delta);  
  
 //Kooordinat ile ölüp ölmediğini anlama  
 if(player.b2body.getPosition().y<0){  
 if(player.b2body.getPosition().x >60){  
 checkPoint= true;  
 }  
 if(checkPoint){  
 player.checkpointDefineCharacter();  
 }  
 else player.defineCharacter();  
 }  
  
 //Oyunu bitirme  
 if(player.b2body.getPosition().x>124){  
 game.changeScreen(DesertGame.*MENU*);  
 player.defineCharacter();  
 }  
 }  
  
 @Override  
 public void render(float delta) {  
 update(delta);  
 // Harita ve kamera güncellenir  
 Gdx.*gl*.glClearColor(0, 0, 0, 1);  
 Gdx.*gl*.glClear(GL20.*GL\_COLOR\_BUFFER\_BIT*);  
 renderer.render();  
  
  
 // Karakter animasyonları için gerekli kodlar  
 game.batch.setProjectionMatrix(gameCam.combined);  
 game.batch.begin();  
 player.draw(game.batch);  
 game.batch.end();  
 }  
  
  
  
 @Override  
 public void resize(int width, int height) {  
 gamePort.update(width, height);  
 }  
  
 @Override  
 public void pause() {  
  
 }  
  
 @Override  
 public void resume() {  
  
 }  
  
 @Override  
 public void hide() {  
  
 }  
  
 @Override  
 public void dispose() {  
 }  
}

MENUSCREEN

package com.game.desert.Screens;  
  
import com.badlogic.gdx.Screen;  
  
import com.badlogic.gdx.Gdx;  
import com.badlogic.gdx.graphics.Color;  
import com.badlogic.gdx.graphics.GL20;  
import com.badlogic.gdx.graphics.Texture;  
import com.badlogic.gdx.graphics.g2d.BitmapFont;  
import com.badlogic.gdx.graphics.g2d.TextureRegion;  
import com.badlogic.gdx.scenes.scene2d.InputEvent;  
import com.badlogic.gdx.scenes.scene2d.InputListener;  
import com.badlogic.gdx.scenes.scene2d.Stage;  
import com.badlogic.gdx.scenes.scene2d.ui.\*;  
import com.badlogic.gdx.scenes.scene2d.utils.TextureRegionDrawable;  
import com.badlogic.gdx.utils.Align;  
import com.badlogic.gdx.utils.viewport.FitViewport;  
import com.game.desert.DesertGame;  
  
public class MenuScreen implements Screen {  
 private DesertGame parent;  
 private Stage stage;  
 private Label label;  
 private Texture background;  
 public MenuScreen(DesertGame game){  
 //Ekranın tanımlanması.  
 parent = game;  
 stage = new Stage(new FitViewport(DesertGame.*WIDTH*,DesertGame.*HEIGHT*));  
 Gdx.*input*.setInputProcessor(stage);  
  
 background = new Texture("ArkaPlan/ArkaPlan.png");  
  
 //Tüm işlemlerimi içeren taslağın oluşturulması.  
 Table table = new Table();  
 table.setFillParent(true);  
 table.setDebug(true);  
 stage.addActor(table);  
 table.setBackground(new TextureRegionDrawable(new TextureRegion(new Texture("ArkaPlan/ArkaPlan.png"))));  
  
  
 // Metnin boyutunu, fontunu ve renklerini ayarlama  
 BitmapFont font = new BitmapFont();  
 font.getData().setScale(6);  
 Label.LabelStyle style = new Label.LabelStyle(font, Color.*WHITE*);  
 label = new Label("DESERT GAME", style);  
 label.setPosition(850, 100, Align.*center*); // Metnin konumunu ayarlama  
 stage.addActor(label);  
  
 Skin skin = new Skin(Gdx.*files*.internal("skin/uiskin.json"));  
  
 // Butonların oluşturulması.  
 TextButton playGame = new TextButton("OYNA", skin);  
 TextButton preferences = new TextButton("AYARLAR", skin);  
 TextButton exit = new TextButton("CIKIS", skin);  
  
  
 playGame.setSize(160, 90);  
 preferences.setSize(160, 90);  
 exit.setSize(160, 90);  
  
  
 // Butonları ekrana ekler.  
 stage.addActor(playGame);  
 stage.addActor(preferences);  
 stage.addActor(exit);  
  
 // Butonların pozisyonlarını belirler  
 playGame.setPosition(100, 400);  
 preferences.setPosition(100, 250);  
 exit.setPosition(100, 100);  
  
  
 //Butonlara basıldığında algılanmasını sağlar.  
 playGame.addListener(new InputListener() {  
 @Override  
 public boolean touchDown(InputEvent event, float x, float y, int pointer, int button) {  
 parent.changeScreen(DesertGame.*APPLICATION*);  
 return true;  
 }  
 });  
  
 preferences.addListener(new InputListener() {  
 @Override  
 public boolean touchDown(InputEvent event, float x, float y, int pointer, int button) {  
 parent.changeScreen(DesertGame.*PREFERENCES*);  
 return true;  
 }  
 });  
  
 exit.addListener(new InputListener() {  
 @Override  
 public boolean touchDown(InputEvent event, float x, float y, int pointer, int button) {  
 Gdx.*app*.exit();  
 return true;  
 }  
 });  
  
 stage.act(Math.*min*(Gdx.*graphics*.getDeltaTime(), 1 / 30f));  
 stage.draw();  
 }  
  
 @Override  
 public void show() {  
 }  
  
  
 @Override  
 public void render(float delta) {  
 //Ekranı tazeler.  
 Gdx.*gl*.glClearColor(0f, 0f, 0f, 1);  
 Gdx.*gl*.glClear(GL20.*GL\_COLOR\_BUFFER\_BIT*);  
  
 stage.act(Math.*min*(Gdx.*graphics*.getDeltaTime(), 1 / 30f));  
 stage.draw();  
 }  
  
 @Override  
 public void resize(int width, int height) {  
 stage.getViewport().update(width, height, true);  
 }  
  
 @Override  
 public void pause() {  
  
 }  
  
 @Override  
 public void resume() {  
  
 }  
  
  
 @Override  
 public void hide() {  
  
 }  
  
 @Override  
 public void dispose() {  
 stage.dispose();  
 }  
}

PAUSESCREEN

package com.game.desert.Screens;  
  
import com.badlogic.gdx.Gdx;  
import com.badlogic.gdx.Screen;  
import com.badlogic.gdx.graphics.Color;  
import com.badlogic.gdx.graphics.GL20;  
import com.badlogic.gdx.graphics.Texture;  
import com.badlogic.gdx.graphics.g2d.BitmapFont;  
import com.badlogic.gdx.graphics.g2d.TextureRegion;  
import com.badlogic.gdx.scenes.scene2d.InputEvent;  
import com.badlogic.gdx.scenes.scene2d.InputListener;  
import com.badlogic.gdx.scenes.scene2d.Stage;  
import com.badlogic.gdx.scenes.scene2d.ui.\*;  
import com.badlogic.gdx.scenes.scene2d.utils.TextureRegionDrawable;  
import com.badlogic.gdx.utils.Align;  
import com.badlogic.gdx.utils.viewport.FitViewport;  
import com.game.desert.DesertGame;  
  
public class PauseScreen implements Screen {  
 //Burası menü ekranının aynısı  
 //Normalde menü ekranına dönemezken bu sınıf sayesinde dönebiliyoruz.  
 private DesertGame parent;  
 private Stage stage;  
 private Label label;  
 private Texture background;  
 public PauseScreen(DesertGame game){  
  
 parent = game;  
 stage = new Stage(new FitViewport(DesertGame.*WIDTH*,DesertGame.*HEIGHT*));  
 Gdx.*input*.setInputProcessor(stage);  
  
 background = new Texture("ArkaPlan/ArkaPlan.png");  
  
  
 Table table = new Table();  
 table.setFillParent(true);  
 table.setDebug(true);  
 stage.addActor(table);  
 table.setBackground(new TextureRegionDrawable(new TextureRegion(new Texture("ArkaPlan/ArkaPlan.png"))));  
  
  
 // Metnin boyutunu, fontunu ve renklerini ayarlama  
 BitmapFont font = new BitmapFont();  
 font.getData().setScale(6);  
 Label.LabelStyle style = new Label.LabelStyle(font, Color.*WHITE*);  
 label = new Label("DESERT GAME", style);  
 label.setPosition(850, 100, Align.*center*); // Metnin konumunu ayarlama  
 stage.addActor(label);  
  
 Skin skin = new Skin(Gdx.*files*.internal("skin/uiskin.json"));  
  
  
 TextButton playGame = new TextButton("OYNA", skin);  
 TextButton preferences = new TextButton("AYARLAR", skin);  
 TextButton exit = new TextButton("CIKIS", skin);  
  
  
 playGame.setSize(160, 90);  
 preferences.setSize(160, 90);  
 exit.setSize(160, 90);  
  
  
 stage.addActor(playGame);  
 stage.addActor(preferences);  
 stage.addActor(exit);  
  
  
 playGame.setPosition(100, 400);  
 preferences.setPosition(100, 250);  
 exit.setPosition(100, 100);  
  
  
 playGame.addListener(new InputListener() {  
 @Override  
 public boolean touchDown(InputEvent event, float x, float y, int pointer, int button) {  
 parent.changeScreen(DesertGame.*APPLICATION*);  
 return true;  
 }  
 });  
  
 preferences.addListener(new InputListener() {  
 @Override  
 public boolean touchDown(InputEvent event, float x, float y, int pointer, int button) {  
 parent.changeScreen(DesertGame.*PREFERENCES*);  
 return true;  
 }  
 });  
  
 exit.addListener(new InputListener() {  
 @Override  
 public boolean touchDown(InputEvent event, float x, float y, int pointer, int button) {  
 Gdx.*app*.exit();  
 return true;  
 }  
 });  
  
 stage.act(Math.*min*(Gdx.*graphics*.getDeltaTime(), 1 / 30f));  
 stage.draw();  
  
  
 }  
  
 @Override  
 public void show() {  
 }  
  
  
 @Override  
 public void render(float delta) {  
 Gdx.*gl*.glClearColor(0f, 0f, 0f, 1);  
 Gdx.*gl*.glClear(GL20.*GL\_COLOR\_BUFFER\_BIT*);  
  
 stage.act(Math.*min*(Gdx.*graphics*.getDeltaTime(), 1 / 30f));  
 stage.draw();  
  
  
 }  
  
 @Override  
 public void resize(int width, int height) {  
 stage.getViewport().update(width, height, true);  
 }  
  
 @Override  
 public void pause() {  
  
 }  
  
 @Override  
 public void resume() {  
  
 }  
  
  
 @Override  
 public void hide() {  
  
 }  
  
 @Override  
 public void dispose() {  
 stage.dispose();  
 }  
}

PREFERENCESSCREEN

package com.game.desert.Screens;  
  
import com.badlogic.gdx.Gdx;  
import com.badlogic.gdx.Screen;  
import com.badlogic.gdx.audio.Music;  
import com.badlogic.gdx.graphics.GL20;  
import com.badlogic.gdx.graphics.Texture;  
import com.badlogic.gdx.graphics.g2d.TextureRegion;  
import com.badlogic.gdx.scenes.scene2d.\*;  
import com.badlogic.gdx.scenes.scene2d.ui.\*;  
import com.badlogic.gdx.scenes.scene2d.utils.TextureRegionDrawable;  
import com.badlogic.gdx.utils.viewport.FitViewport;  
import com.game.desert.DesertGame;  
  
public class PreferencesScreen implements Screen{  
  
 private DesertGame parent;  
 private MenuScreen menuScreen;  
 private Stage stage;  
 private Label titleLabel;  
 private Label volumeMusicLabel;  
 private Label musicOnOffLabel;  
 private Music music;  
  
  
  
 public PreferencesScreen(DesertGame game){  
 //Ayarlar için bir taslak oluşturur.  
 parent = game;  
 stage = new Stage(new FitViewport(DesertGame.*WIDTH*,DesertGame.*HEIGHT*));  
 }  
  
 @Override  
 public void show() {  
 stage.clear();  
 Gdx.*input*.setInputProcessor(stage);  
  
 Table table = new Table();  
 table.setFillParent(true);  
 table.setBackground(new TextureRegionDrawable(new TextureRegion(new Texture("ArkaPlan/ArkaPlan.png"))));  
 stage.addActor(table);  
  
  
 Skin skin = new Skin(Gdx.*files*.internal("skin/uiskin.json"));  
  
 // Müzik sesi ayarlama  
 final Slider volumeMusicSlider = new Slider(0f, 1f, 0.1f, false, skin);  
 volumeMusicSlider.setValue(parent.getPreferences().getMusicVolume());  
 volumeMusicSlider.addListener(new EventListener() {  
 @Override  
 public boolean handle(Event event) {  
 parent.getPreferences().setMusicVolume(volumeMusicSlider.getValue());  
 if(parent.getPreferences().isMusicEnabled()){  
 music = DesertGame.*manager*.get("Music/arkaplan\_1.ogg", Music.class);  
 music.setLooping(true);  
 music.setVolume(volumeMusicSlider.getValue());  
 music.play();  
 }  
 else{  
 music = DesertGame.*manager*.get("Music/arkaplan\_1.ogg", Music.class);  
 music.setLooping(true);  
 music.setVolume(volumeMusicSlider.getValue());  
 music.stop();  
 }  
  
 return false;  
 }  
 });  
  
 // Müzik açma kapama  
 final CheckBox musicCheckbox = new CheckBox(null, skin);  
 musicCheckbox.setChecked(parent.getPreferences().isMusicEnabled());  
 musicCheckbox.addListener(new EventListener() {  
 @Override  
 public boolean handle(Event event) {  
 boolean enabled = musicCheckbox.isChecked();  
 parent.getPreferences().setMusicEnabled(enabled);  
 return false;  
 }  
 });  
  
  
 // Geri dönme butonu  
 final TextButton backButton = new TextButton("Geri", skin);  
 backButton.addListener(new InputListener() {  
 @Override  
 public boolean touchDown(InputEvent event, float x, float y, int pointer, int button) {  
 parent.changeScreen(DesertGame.*MENU*);  
 menuScreen=new MenuScreen(parent);  
 return true;  
 }  
 });  
 //Butonları ekleme  
 titleLabel = new Label( "Ayarlar", skin );  
 volumeMusicLabel = new Label( "Muzik seviyesi", skin );  
 musicOnOffLabel = new Label( "Muzik", skin );  
  
 table.add(titleLabel).colspan(2);  
 table.row().pad(10,0,0,10);  
 table.add(volumeMusicLabel).left();  
 table.add(volumeMusicSlider);  
 table.row().pad(10,0,0,10);  
 table.add(musicOnOffLabel).left();  
 table.add(musicCheckbox);  
 table.row().pad(10,0,0,10);  
  
 table.add(backButton).colspan(2);  
  
 }  
  
 @Override  
 public void render(float delta) {  
 //Ekranı temizler  
 Gdx.*gl*.glClearColor(0f, 0f, 0f, 1);  
 Gdx.*gl*.glClear(GL20.*GL\_COLOR\_BUFFER\_BIT*);  
  
 stage.act(Math.*min*(Gdx.*graphics*.getDeltaTime(), 1 / 30f));  
 stage.draw();  
  
 }  
  
 @Override  
 public void resize(int width, int height) {  
 stage.getViewport().update(width, height, true);  
 }  
  
 @Override  
 public void pause() {  
 // *TODO Auto-generated method stub* }  
  
 @Override  
 public void resume() {  
 // *TODO Auto-generated method stub* }  
  
 @Override  
 public void hide() {  
 // *TODO Auto-generated method stub* }  
  
 @Override  
 public void dispose() {  
 // *TODO Auto-generated method stub* }  
  
}

APP REFERENCES

package com.game.desert;  
  
import com.badlogic.gdx.Gdx;  
import com.badlogic.gdx.Preferences;  
  
public class AppPreferences {  
 //Ayarların kaydedilmesini sağlar.  
 private static final String *PREF\_MUSIC\_VOLUME* = "volume";  
 private static final String *PREF\_MUSIC\_ENABLED* = "music.enabled";  
 private static final String *PREFS\_NAME* = "desertGame";  
  
 protected Preferences getPrefs() {  
 return Gdx.*app*.getPreferences(*PREFS\_NAME*);  
 }  
 public boolean isMusicEnabled() {  
 return getPrefs().getBoolean(*PREF\_MUSIC\_ENABLED*, true);  
 }  
  
 public void setMusicEnabled(boolean musicEnabled) {  
 getPrefs().putBoolean(*PREF\_MUSIC\_ENABLED*, musicEnabled);  
 getPrefs().flush();  
 }  
  
 public float getMusicVolume() {  
 return getPrefs().getFloat(*PREF\_MUSIC\_VOLUME*, 0.5f);  
 }  
  
 public void setMusicVolume(float volume) {  
 getPrefs().putFloat(*PREF\_MUSIC\_VOLUME*, volume);  
 getPrefs().flush();  
 }  
}

CHARACTER

package com.game.desert.Sprites;  
  
import com.badlogic.gdx.graphics.g2d.Animation;  
import com.badlogic.gdx.graphics.g2d.Sprite;  
import com.badlogic.gdx.graphics.g2d.TextureRegion;  
import com.badlogic.gdx.physics.box2d.\*;  
import com.badlogic.gdx.utils.Array;  
import com.game.desert.DesertGame;  
import com.game.desert.Screens.PlayScreen;  
  
//Karakterin oluşturulması  
public class Character extends Sprite {  
 //Aşağıdaki durumlar karakterin haraket durumlarını belirtir.  
 public enum State{*FALLING*, *JUMPING*, *STANDING*, *RUNNING*}  
 public State currentState;  
 public State previousState;  
 private Animation <TextureRegion> characterRun;  
 private Animation <TextureRegion> characterJump;  
 private boolean runningRight; //Sola koşarken sola dönmesini sağlıyor  
 private float stateTimer;  
 public World world;  
 public Body b2body;  
 private TextureRegion characterStand;  
  
 public Character(World world, PlayScreen screen) {  
 //Karakterin resmini tanımlar  
 super(screen.getAtlas().findRegion("Girl"));  
 this.world = world;  
  
 currentState= State.*STANDING*;  
 previousState= State.*STANDING*;  
 stateTimer= 0;  
 runningRight= true;  
  
 // Animasyon görselleini içeren dizi  
 Array<TextureRegion> frames = new Array<TextureRegion>();  
  
  
 // Koşma animasyonu için for döngüsü  
 for(int i= 0 ; i<6 ; i++)  
 frames.add(new TextureRegion(getTexture(),i\*48 ,18,30,48));  
  
 characterRun= new Animation<TextureRegion>(0.1f,frames);  
 frames.clear();  
  
  
 // Zıplama animasyonu için for döngüsü  
 for(int i= 3 ; i<4 ; i++ )  
 frames.add(new TextureRegion(getTexture(),143 ,18,30,48));  
  
 characterJump= new Animation<TextureRegion>(0.1f,frames);  
  
 defineCharacter();  
  
 // Karakterin hareketsiz halinin görüntüsünün kaçıncı pixellerden alınacağı belirleniyor  
 characterStand= new TextureRegion(getTexture(), 290, 18, 30, 48);  
 // Karakterin boyutu  
 setBounds(0, 0, 100/DesertGame.*PPM* , 150/DesertGame.*PPM*);  
 setRegion(characterStand);  
 }  
  
  
  
  
 public void update(float dt){  
 setPosition(b2body.getPosition().x - getWidth()/2, b2body.getPosition().y - getHeight()/2);  
 setRegion(getFrame(dt));  
 }  
  
  
 public TextureRegion getFrame(float dt){  
 //Karakterin anlık durumunu belirler.  
 currentState= getState();  
  
 TextureRegion region;  
 switch (currentState){  
 case *JUMPING*:  
 region = (TextureRegion) characterJump.getKeyFrame(stateTimer);  
 break;  
 case *RUNNING*:  
 region = (TextureRegion) characterRun.getKeyFrame(stateTimer, true);  
 break;  
 case *FALLING*:  
 case *STANDING*:  
 default:  
 region= characterStand;  
 break;  
 }  
  
 // Karakterin sağa veya sola koşarken koştuğu yöne dönmesini sağlayan if yapısı  
 if((b2body.getLinearVelocity().x < 0 || !runningRight ) && !region.isFlipX()){  
  
 region.flip(true,false);  
 runningRight= false;  
 }  
  
 else if((b2body.getLinearVelocity().x > 0 || runningRight ) && region.isFlipX()) {  
  
 region.flip(true, false);  
 runningRight = true;  
 }  
  
 stateTimer= currentState == previousState ? stateTimer + dt : 0;  
 previousState = currentState;  
 return region;  
 }  
  
 public State getState(){  
 //Karakterin durumunu verir.  
 if (b2body.getLinearVelocity().y > 0 || (b2body.getLinearVelocity().y < 0 && previousState==State.*JUMPING*)) return State.*JUMPING*;  
 else if (b2body.getLinearVelocity().y < 0) return State.*FALLING*;  
 else if(b2body.getLinearVelocity().x != 0) return State.*RUNNING*;  
 else return State.*STANDING*;  
 }  
 public void defineCharacter() {  
 //Karakterin vücudunun oluşturulması  
 BodyDef bodyDef = new BodyDef();  
 bodyDef.position.set((1/ DesertGame.*PPM*)+9,300/DesertGame.*PPM*); //Karakterin doğma yeri  
 bodyDef.type = BodyDef.BodyType.*DynamicBody*;  
 b2body = world.createBody(bodyDef);  
  
 //Karakterin temel yapısının şekli(çember)  
 FixtureDef fixtureDef = new FixtureDef();  
 CircleShape shape = new CircleShape();  
 shape.setRadius(25/DesertGame.*PPM*);  
 fixtureDef.shape = shape;  
 b2body.createFixture(fixtureDef);  
 }  
  
 public void checkpointDefineCharacter(){  
 //Kayıt yeri  
 BodyDef bodyDef = new BodyDef();  
 bodyDef.position.set((1/ DesertGame.*PPM*)+60,300/DesertGame.*PPM*); //Karakterin doğma yeri  
 bodyDef.type = BodyDef.BodyType.*DynamicBody*;  
 b2body = world.createBody(bodyDef);  
  
 //Karakterin temel yapısının şekli(çember)  
 FixtureDef fixtureDef = new FixtureDef();  
 CircleShape shape = new CircleShape();  
 shape.setRadius(25/DesertGame.*PPM*);  
 fixtureDef.shape = shape;  
 b2body.createFixture(fixtureDef);  
 }  
}