CODENAME: space-simulator

HIGH-LEVEL CODE DOCUMENTATION

# Java-packages

Проект состоит из трех java-пакетов:  
1. com.nikarus.spacesimulator – в нем хранятся системные классы;

2. com.nikarus.spacesimulator.shipparts – в нем хранятся классы деталей коробля(двигатель, оружие и т.п.);

3. com.nikarus.spacesimulator.spacecraft – в нем хранятся классы собранных объектов космоса – начиная от астеройда, кончая самими кораблями.

# com.nikarus.spacesimulator

Классы:

1. SpaceSimulatorActivity – главный класс, с которого все начинается.
2. Touch – Класс для упрощения работы с тачами. Вероятно всю тач схему нужно будет переделать, так что он может уйти.
3. Map – класс который отвечает за создание разных карт. По хорошему карты нужно хранить в xml-файлах. А сам этот класс вынести из этого пакета.

# com.nikarus.spacesimulator.shipparts

Основные детали, из которых состоит корабль это:

1. ShipPart – любая деталь корабля, которая является пассивной;
2. ShipEngine – наследуется от ShipPart. Представляет из себя двигатель;
3. Weapon – наследуется от ShipPart. Является оружием.
4. Также там есть класс Missile – ракета. Наследуется от ShipEngine.

# com.nikarus.spacesimulator.spacecraft

1. Ship - основной класс, от которого наследуются все корабли;
2. Ship\_1b1f2l2r – корабль. «1b1f2l2r» означает «1 back engine, 1 front engine, 2 left engines and 2 right engines»
3. Ship\_2b1f – 2 back and 1 front engine;
4. Ship\_2b2f – 2 back and 2 front engines;
5. Obstacle – любое препятствие. Например астероид.

# Ship creation example

Пример основан на классе Ship\_2b1f.

Чтобы создать корабль необходимо:

1. Создать класс и пронаследовать его от класса Ship:

**public** **class** Ship\_2b1f **extends** Ship

1. Инициализировать атлас текстур и вызвать loadCommonData (процедура класса Ship):

**private** **static** BitmapTextureAtlas *mBitmapTextureAtlas* = **null**;

**public** Ship\_2b1f(….) {

**if** (*mBitmapTextureAtlas*==**null**) *mBitmapTextureAtlas* = **new** BitmapTextureAtlas(context.getTextureManager(), 256, 256);

loadCommonData(*mBitmapTextureAtlas*, context, engine, scene, physicsWorld, **this**, isHumanControlled, isCameraTraced);

1. Загрузить текстуры из файлов (на данный момент класс Ship является агрегатором всех текстур)

**if** (Ship.*sEngineFireTextureRegion* == **null**) Ship.*sEngineFireTextureRegion* = BitmapTextureAtlasTextureRegionFactory.*createTiledFromAsset*(*mBitmapTextureAtlas*, context, "EngineFireCropped160.png", 0, 0, 5, 1);

**..........**

engine.getTextureManager().loadTexture(*mBitmapTextureAtlas*);

1. Создать тело корабля, к которому будут крепиться все двигатели и оружия:

**this**.mShipFixtureDef = PhysicsFactory.*createFixtureDef*(1f, 0.1f, 0.2f);

**float** width=Ship.*sShipBodyTextureRegion*.getWidth() / *PIXEL\_TO\_METER\_RATIO\_DEFAULT*;

**float** height=Ship.*sShipBodyTextureRegion*.getHeight() / *PIXEL\_TO\_METER\_RATIO\_DEFAULT*;

**final** Vector2[] ShipBodyVertices = {

**new** Vector2(-0.36452f\*width, -0.32476f\*height),

**new** Vector2(-0.11070f\*width, -0.51265f\*height),

**new** Vector2(+0.10250f\*width, -0.51265f\*height),

**new** Vector2(+0.34109f\*width, -0.32476f\*height),

**new** Vector2(+0.49338f\*width, +0.06605f\*height),

**new** Vector2(+0.43754f\*width, +0.26897f\*height),

**new** Vector2(-0.45082f\*width, +0.33661f\*height),

**new** Vector2(-0.50158f\*width, +0.05102f\*height)

};

**this**.mShipBodySprite = **new** Sprite(shipX, shipY, Ship.*sShipBodyTextureRegion*, context.getVertexBufferObjectManager());

**this**.mShipBody = PhysicsFactory.*createPolygonBody*(physicsWorld, **this**.mShipBodySprite, ShipBodyVertices, BodyType.*DynamicBody*, **this**.mShipFixtureDef);

scene.attachChild(**this**.mShipBodySprite);

1. Создать двигатели:

width=Ship.*sLeftEngineTextureRegion*.getWidth() / *PIXEL\_TO\_METER\_RATIO\_DEFAULT*;

height=Ship.*sLeftEngineTextureRegion*.getHeight() / *PIXEL\_TO\_METER\_RATIO\_DEFAULT*;

**final** Vector2[] leftEngineVertices = {

**new** Vector2(-0.30167f\*width, -0.49382f\*height),

**new** Vector2(+0.43033f\*width, +0.03029f\*height),

**new** Vector2(+0.22633f\*width, +0.53324f\*height),

**new** Vector2(-0.50567f\*width, +0.22618f\*height)

};

……..

ShipEngine ShipEngine [] = {**new** ShipEngine(context, **this**, 25, -10, (**float**)Math.*PI*, frontEngineForce, frontEngineStrength, **this**.mShipBody, **this**.mShipBody, **this**.mShipBodySprite, frontEngineVertices, scene, physicsWorld, *sEngineFireTextureRegion*, 0, -Ship.*sFrontEngineTextureRegion*.getHeight(), 1, Ship.*sFrontEngineTextureRegion*, **this**.mShipFixtureDef, **true**),

**new** ShipEngine(…),

**new** ShipEngine(…))};

**this**.mShipEngine=ShipEngine;

1. Создать оружия(пулемет и ракетницу):

width=Ship.*sRocketTextureRegion*.getWidth() / *PIXEL\_TO\_METER\_RATIO\_DEFAULT*;

height=Ship.*sRocketTextureRegion*.getHeight() / *PIXEL\_TO\_METER\_RATIO\_DEFAULT*;

**final** Vector2[] missileVertices = {

**new** Vector2(-0.59465f\*width, -0.47054f\*height),

**new** Vector2(-0.02973f\*width, -0.50988f\*height),

**new** Vector2(+0.50514f\*width, -0.47153f\*height)

};

width=Ship.*sBulletTextureRegion*.getWidth() / *PIXEL\_TO\_METER\_RATIO\_DEFAULT*;

height=Ship.*sBulletTextureRegion*.getHeight() / *PIXEL\_TO\_METER\_RATIO\_DEFAULT*;

**final** Vector2[] bulletVertices = {

**new** Vector2(-0.23070f\*width, -0.48555f\*height),

**new** Vector2(-0.02071f\*width, -0.50780f\*height),

**new** Vector2(+0.18164f\*width, -0.48555f\*height)

};

width=Ship.*sShip2ChainGunTextureRegion*.getWidth() / *PIXEL\_TO\_METER\_RATIO\_DEFAULT*;

height=Ship.*sShip2ChainGunTextureRegion*.getHeight() / *PIXEL\_TO\_METER\_RATIO\_DEFAULT*;

**final** Vector2[] chainGunVertices = {

**new** Vector2(-0.38667f\*width, -0.51030f\*height),

**new** Vector2(+0.45316f\*width, -0.50284f\*height),

**new** Vector2(+0.01710f\*width, +0.49601f\*height)

};

width=Ship.*sShip2RocketLauncherTextureRegion*.getWidth() / *PIXEL\_TO\_METER\_RATIO\_DEFAULT*;

height=Ship.*sShip2RocketLauncherTextureRegion*.getHeight() / *PIXEL\_TO\_METER\_RATIO\_DEFAULT*;

**final** Vector2[] rocketLauncherVertices = {

**new** Vector2(-0.39978f\*width, -0.46213f\*height),

**new** Vector2(+0.25917f\*width, -0.46213f\*height),

**new** Vector2(-0.08969f\*width, +0.45405f\*height)

};

mWeapon.add(**new** Weapon(**this**, 90, 5, 93, -12, 0, 3000, **true**, 100, 2000, 10f, **this**.mShipBody, **this**.mShipBodySprite, rocketLauncherVertices, missileVertices, scene, physicsWorld, *sEngineFireTextureRegion*, 0, Ship.*sRocketTextureRegion*.getHeight()-3, 1, *sShip2RocketLauncherTextureRegion*, Ship.*sRocketTextureRegion*, **this**.mShipFixtureDef, **this**.mShipFixtureDef));

mWeapon.add(**new** Weapon(**this**, 10, -2, 10, -4, 0, 2000, **false**, 0.5f, 300, 10f, **this**.mShipBody, **this**.mShipBodySprite, chainGunVertices, bulletVertices, scene, physicsWorld, **null**, 0, 0, 0, *sShip2ChainGunTextureRegion*, Ship.*sBulletTextureRegion*, **this**.mShipFixtureDef, **this**.mShipFixtureDef));

1. Отправить все созданное в мир сей:

physicsWorld.registerPhysicsConnector(**new** PhysicsConnector(**this**.mShipBodySprite, **this**.mShipBody, **true**, **true**));

1. Повернуть его в нужную сторону:

**this**.mShipBody.setTransform((mShipBodySprite.getX()+mShipBodySprite.getRotationCenterX())/*PIXEL\_TO\_METER\_RATIO\_DEFAULT*,(mShipBodySprite.getY()+mShipBodySprite.getRotationCenterY())/*PIXEL\_TO\_METER\_RATIO\_DEFAULT*, rotation\*(**float**)Math.*PI*/180);

1. Перегрузить процедуру control, где указать какие области экрана будут отвечать за какие двигатели и оружие:

@Override

**public** **void** control(Touch [] touches) {

**if** (!mIsHumanControlled) **return**;

**if** (*isAreaTouched*(0,240,150,480,touches)) {//LEFT-BOTTOM

mShipEngine[2].engineStart();

} **else** {

mShipEngine[2].engineStop();

}

**if** (*isAreaTouched*(0,0,150,240,touches) /\*LEFT-TOP\*/|| *isAreaTouched*(650,0,800,240,touches)/\*RIGHT-TOP\*/) {

mShipEngine[0].engineStart();

} **else** {

mShipEngine[0].engineStop();

}

**if** (*isAreaTouched*(650,240,800,480,touches)) {//RIGHT-BOTTOM

mShipEngine[1].engineStart();

} **else** {

mShipEngine[1].engineStop();

}

**if** (*isAreaTouched*(150,0,300,240,touches) || *isAreaTouched*(500,0,650,240,touches)) {

mWeapon.get(0).weaponStartShooting();

} **else** {

mWeapon.get(0).weaponStopShooting();

}

**if** (*isAreaTouched*(150,240,300,480,touches) || *isAreaTouched*(500,240,650,480,touches)) {

mWeapon.get(1).weaponStartShooting();

} **else** {

mWeapon.get(1).weaponStopShooting();

}

}

1. Чтобы поместить корабль на карту, необходимо в процедуре создания соответствующей карты вызвать его конструктор:

**new** Ship\_2b1f(context, mEngine, mPhysicsWorld, mScene, **false**, **false**, 1000, 600, 0, 40, 40, 40, 50f, 50f, 80f);

1. Также нужно убедиться, что SimpleBaseGameActivity. onCreateScene создается именно та карта ☺

Map map=**new** Map();

map.CreateMetalMap(mEngine, mScene, mPhysicsWorld, **this**);