Multimedia-Programmierung
Übung 4

Ludwig-Maximilians-Universität München
Sommersemester 2016
Today
Overview – Alternative Engines

libGDX
https://libgdx.badlogicgames.com/

Pygame
http://www.pygame.org/hifi.html

Unity
https://unity3d.com/

SpriteKit for Apple platforms:

Corona Labs
https://coronalabs.com/

And many more…
What is Cocos and where does it come from?

- **Cocos2d-x** is an open source, cross-platform game engine

- **Cocos** is a platform to create games, including **Cocos2d-x**, editors, SDK integration

Los Cocos, Argentina
The Platform

Cocos Platform

Cocos Launcher

Scene Editors
- Cocos Studio
- Cocos Creator
- Third Party Editor

SDK Integration
- AnySDK
- SDKBox

Libraries
- SDK Plugins
- Cocos2D-X Extensions
- Cocos2D-X (C++, Lua, JavaScript)

Develop platforms: Win, Mac
Target platforms: Android, iOS, WP, HTML5

http://www.cocos2d-x.org/docs/cocos/cocos/index.html
What is a game engine?

http://www.cocos2d-x.org/docs/programmers-guide/1/index.html

• A piece of software providing common functionality that all games need

• Important components:
  • Renderer
  • 2d/3d graphics
  • Collision detection
  • Physics engine
  • Animations
  • Sound

• Cross-platform: develop once and deploy to multiple platforms.
Cocos street credibility!? 

Badland

Dragon City

Castle Clash

Big Fish Casino

Many more: http://www.cocos2d-x.org/games
Prerequisites

• A working C++ compiler
• Windows: Visual Studio Express
• Mac OS: Xcode
• Python **2.7.x** (version matters!)
• Optional: Android SDK
• And of course: cocos2d-x

http://www.cocos2d-x.org/docs/static-pages/installation.html

http://www.cocos2d-x.org/docs/installation/A/index.html
Installation and Use of Cocos

http://www.cocos2d-x.org/products

Cocos

Cocos is a free and professional game-development toolkit, that enables developers to quickly create game content and remove the tedious work by simplifying it with straightforward GUI editors.

Cocos includes: the Cocos2d-x game engine, a game development environment and project management tool. With this suite, developers can focus on their roles and enjoy a better streamlined workflow. This saves game studios time and money by allowing them to collaborate with ease, and focus on what they do best to achieve better quality and faster turnaround time.

DOWNLOAD
Cocos2d-x basics

Core concepts:
Director, Scene, Node, Sprite, Action
Cocos2d-x basics: Director

- Singleton object, always accessible
- Used to control game flow
Cocos2d-x basics: Scene

- Typical scenes: main menu, levels, game over
- Renders graphical scene objects

Game example: Banana Kong
Cocos2d-x basics: Scenegraph

- Tree structure
- Arranges graphical scene objects in parent-child relationships
- Contains *node* objects

Nodes in this scene?
Cocos2d-x basics: Sprites

- 2D images that can be moved and transformed
- Example: main character, enemies etc…
- Configurable properties: position, rotation, scale, opacity, color
Cocos2d-x basics: Actions

- Action objects make a Node perform a change to its properties over time
- Example: move a sprite from one position to another over a span of time
- Example Actions: MoveBy, Rotate, Scale
- Applicable to all Node objects
- Also available: sequences and spawns of actions
Building a new Project

Using Cocos GUI:
http://www.cocos2d-x.org/docs/cocos/cocos/index.html#using-cocos

Using the command line tool (only for Python 2.7):
http://www.cocos2d-x.org/docs/editors_and_tools/cocosCLTool/index.html
Building a new Project using the GUI
Result
Result

IDE projects (Visual Studio, Xcode etc.)

Your C++ files

Images, Sounds, etc…
Hello World – Auto-generated App
Hello World – HelloWorldScene.cpp

Creating the scene

```cpp
#include "HelloWorldScene.h"
#include "cocostudio/CocoStudio.h"
#include "ui/CocosGUI.h"

USING_NS_CC;

using namespace cocostudio::timeline;

Scene* HelloWorld::createScene()
{
    // 'scene' is an autorelease object
    auto scene = Scene::create();

    // 'layer' is an autorelease object
    auto layer = HelloWorld::create();

    // add layer as a child to scene
    scene->addChild(layer);

    // return the scene
    return scene;
}
```
Hello World – AppDelegate.cpp

```cpp
bool AppDelegate::applicationDidFinishLaunching() {
    // initialize director
    auto director = Director::getInstance();
    auto glview = director->getOpenGLView();
    if(!glview) {
        glview = GLViewImpl::createWithRect("HelloWorld", Rect(0, 0, 960, 640));
        director->setOpenGLView(glview);
    }

director->getOpenGLView()->setDesignResolutionSize(960, 640, ResolutionPolicy::SHOW_ALL);

    // turn on display FPS
    director->setDisplayStats(true);

    // set FPS. the default value is 1.0/60 if you don't call this
    director->setAnimationInterval(1.0 / 60);

    FileUtils::getInstance()->addSearchPath("res");

    // create a scene. it's an autorelease object
    auto scene = HelloWorld::createScene();

    // run
    director->runWithScene(scene);

    return true;
}
```

Using the director to run the game with the scene
Today’s Project: A small game!
Airhockey-Game

3 Classes:
- AppDelegate to run the Game
- GameLayer includes the game function
- GameSprite for the movable objects (players + ball)

Focus on GameLayer
Airhockey-Game: GameLayer

Create init () which includes

- 2 player and 1 ball

```cpp
_player1 = GameSprite::gameSpriteWithFile("res/mallet.png");
_player1->setPosition(Vec2(_screenSize.width * 0.5, _player1->radius() * 2));
_players.pushBack(_player1);
this->addChild(_player1);
```

- EventListener

```cpp
auto listener =EventListenerTouchAllAtOnce::create();
listener->onTouchesBegan = CC_CALLBACK_2(GameLayer::onTouchesBegan, this);
listener->onTouchesMoved = CC_CALLBACK_2(GameLayer::onTouchesMoved, this);
listener->onTouchesEnded = CC_CALLBACK_2(GameLayer::onTouchesEnded, this);
_eventDispatcher->addEventListenerWithSceneGraphPriority(listener, this);
```
Airhockey-Game: GameLayer (2)

Create update()

```cpp
void GameLayer::update(float dt) {
    auto ballNextPosition = _ball->getNextPosition();
    auto ballVector = _ball->getVector();
    ballVector *= 0.98f;
    ballNextPosition.x += ballVector.x;
    ballNextPosition.y += ballVector.y;
    float squared_radii = pow(_player1->radius() + _ball->radius(), 2);
    for (auto player : _players) {
        auto playerNextPosition = player->getNextPosition();
        auto playerVector = player->getVector();
        float diffx = ballNextPosition.x - player->getPositionX();
        float diffy = ballNextPosition.y - player->getPositionY();
        float distance1 = pow(diffx, 2) + pow(diffy, 2);
        float distance2 = pow(_ball->getPositionX() - playerNextPosition.x, 2)
                         + pow(_ball->getPositionY() - playerNextPosition.y, 2);
        if (distance1 <= squared_radii || distance2 <= squared_radii) {
            float mag_ball = pow(ballVector.x, 2) + pow(ballVector.y, 2);
            float mag_player = pow(playerVector.x, 2) + pow(playerVector.y, 2);
            float force = sqrt(mag_ball + mag_player);
            float angle = atan2(diffy, diffx);
            _ball->setVector(x * force * cosine(angle),
                             y * force * sine(angle));
        }
    }
}
```
Airhockey-Game: GameLayer (3)

Define Eventlistener
- onTouchBegan
- onTouchMoved
- onTouchEnded

```cpp
void GameLayer::onTouchesBegan(const std::vector<Touch*> &touches, Event* event)
{
    for (auto touch : touches) {
        if (touch != nullptr) {
            auto tap = touch->getLocation();
            for (auto player : _players) {
                if (player->boundingBox().containsPoint(tap)) {
                    player->setTouch(tap);
                }
            }
        }
    }
}
```
Links

http://www.cocos2d-x.org/wiki/External_Tutorials


http://www.gamefromscratch.com/page/Cocos2d-x-CPP-Game-Programming-Tutorial-Series.aspx