3 Multimedia Programming with C++ and Multimedia Frameworks

3.1 Multimedia Support by Languages and Frameworks
3.2 Introduction to C++
3.3 SFML: Low-Level Multimedia/Game Framework for C++
3.4 Cocos2d-x: High Level Game Engine for C++

Literature:
cocos2d-x.org
History and Status of Cocos2d(-x)

• February 2008, in "Los Cocos" (Argentina):
  – Meeting of Python(!) game developers
  – Ricardo Quesada and others create 2D game engine called "Cocos2d"
• Juli 2008: Apple AppStore for iPhone, iOS SDK
  – Cocos2d rewritten in Objective-C ("cocos2d-iphone")
  – Successful game apps for iPhone based on Cocos2d (e.g. StickWars)
• Various ports and branches of the framework
• November 2010: Cocos2d-x by Zhe Wang (Xiamen, China)
  – Based on C++
  – One codebase for many platforms
• Cocos2d-x is maintained by Chukong Technologies Inc. (Beijing)
  – Latest version 3.10 (Jan 11, 2016)
  – Claimed to be used by more than 400,000 developers worldwide
Basic Concepts: Layer, Scene, Node, Sprite

Layer, Sprite, DrawNode, Scene, …: Subclasses of Node
Basic Concepts: Scene Graph

Scene is a special type of node which is used as root.
QUIZ!

• What are the necessary statements (assignments) to create the tree structure indicated on the last slide?
Steps to Create a Basic Window

- Create a subclass of `Layer` (here: `SlideShowScene`)
  - Define an initialization method `init()`
- Create a new `Scene` object `scene`
- Create a new object of the subclass of layer (`SlideShowScene`)
  - Add it as a child to the `scene` object
  - Hand it over to the engine for running
- Engine calls `init()` method for new object
  - Initialization code
  - Schedule functions and actions to be run
Creating a Standard Code Skeleton

• Using a Python(!)-implemented command for the terminal command line
  
  cocos new -l cpp -p slideshow0
  -d /Users/hussmann/cocostest SlideShow0

• Creates two classes (four files) with a standard structure
• Following code follows this given structure!
Code Skeleton for Slide Show

class SlideShowScene : public cocos2d::Layer {
...
public:
    static Scene* createScene();

    virtual bool init();

    CREATE_FUNC(SlideShowScene); // implement "static create()" method

    void update(float dt);
};

Scene* SlideShowScene::createScene() {
    auto scene = Scene::create();
    auto layer = SlideShowScene::create();
    scene->addChild(layer);
    return scene;
}

bool SlideShowScene::init() {
    ...
Concepts: AppDelegate, Director

Class AppDelegate is home of the framework
- No need to change much in it!
- Contains the code:
  ```cpp
  auto scene = SlideShowScene::createScene();
  director->runWithScene(scene);
  ```

Director object
- Keeps track of basic properties of application (e.g. size, view)
- Provides access to management of control flow:
  e.g. scheduler, event dispatcher, action manager
SlideShowScene: Prelude

```cpp
#include "SlideShowScene.hpp"
#include <array>

using namespace cocos2d;

const int NUM_PICS = 4;  // number of available slides
const int INTERVAL = 4;  // interval for slide change, in seconds
const int BORDER = 30;   // size of framing border in design pixels
const Color4F bkgColor = Color4F(255/255.0, 228/255.0, 95/255.0, 1);   // Background color (yellowish)

std::array<std::string, NUM_PICS> picFiles =
  {"frog.jpg","cows.jpg","elephant.jpg","tiger.jpg");
  // Traditional C style would be
  // std::string gPicFiles[NUM_PICS{"frog.jpg","cows.jpg",
  // "elephant.jpg","tiger.jpg");

Size visibleSize;
Size imageSpriteSize;
float originXImg;
float originYImg;
```
QUIZ!

• What is Color4F?
• What is the meaning of a value of 1.0 on fourth position?
bool SlideShowScene::init() {
    // super init first
    if (!Layer::init()) {
        return false;
    }

    // Initialize variables
    visibleSize = Director::getInstance()->getVisibleSize();
    Vec2 origin = Director::getInstance()->getVisibleOrigin();
    picIndex = 0;

    // Create DrawNode for image frame rectangle
    imageFrame = DrawNode::create();
    this->addChild(imageFrame);

    // Create sprite for images
    imageSprite = Sprite::create();
    this->addChild(imageSprite);

    // Preload first image and determine image size
    imageSprite->setTexture(picFiles[picIndex]);
    imageSpriteSize = imageSprite->getContentSize();
SlideShowScene: Initialization (2)

... // Position the sprite at the center of the screen
originXImg = origin.x + visibleSize.width/2;
originYImg = origin.y + visibleSize.height/2;
imageSprite->setPosition(Vec2(originXImg, originYImg));

// Draw the border, based on the coordinates and sizes we have computed
imageFrame->drawSolidRect(
    Vec2(originXImg - imageSpriteSize.width/2 - BORDER,
         originYImg - imageSpriteSize.height/2 - BORDER),
    Vec2(originXImg + imageSpriteSize.width/2 + BORDER,
         originYImg + imageSpriteSize.height/2 + BORDER), bkgColor);

// schedule the update function to be called every INTERVAL seconds
this->schedule(schedule_selector(SlideShowScene::update), INTERVAL);
return true;

void SlideShowScene::update (float dt) {
    CCLOG("update called, dt = %f, picIndex = %i", dt, picIndex);
    // Increase picIndex and set image according to picIndex
    picIndex = (picIndex+1) % NUM_PICS;
    imageSprite->setTexture(picFiles[picIndex]);
}