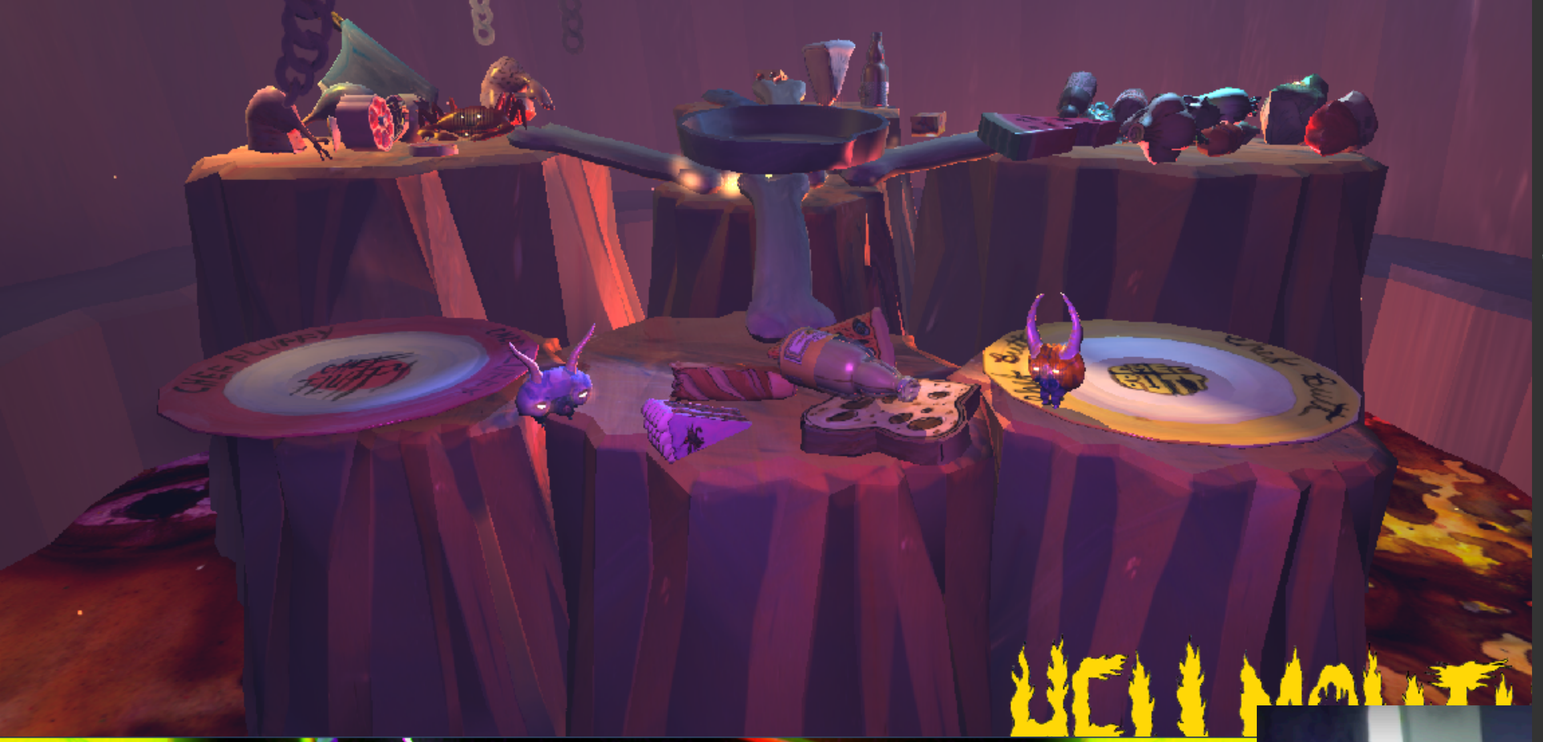
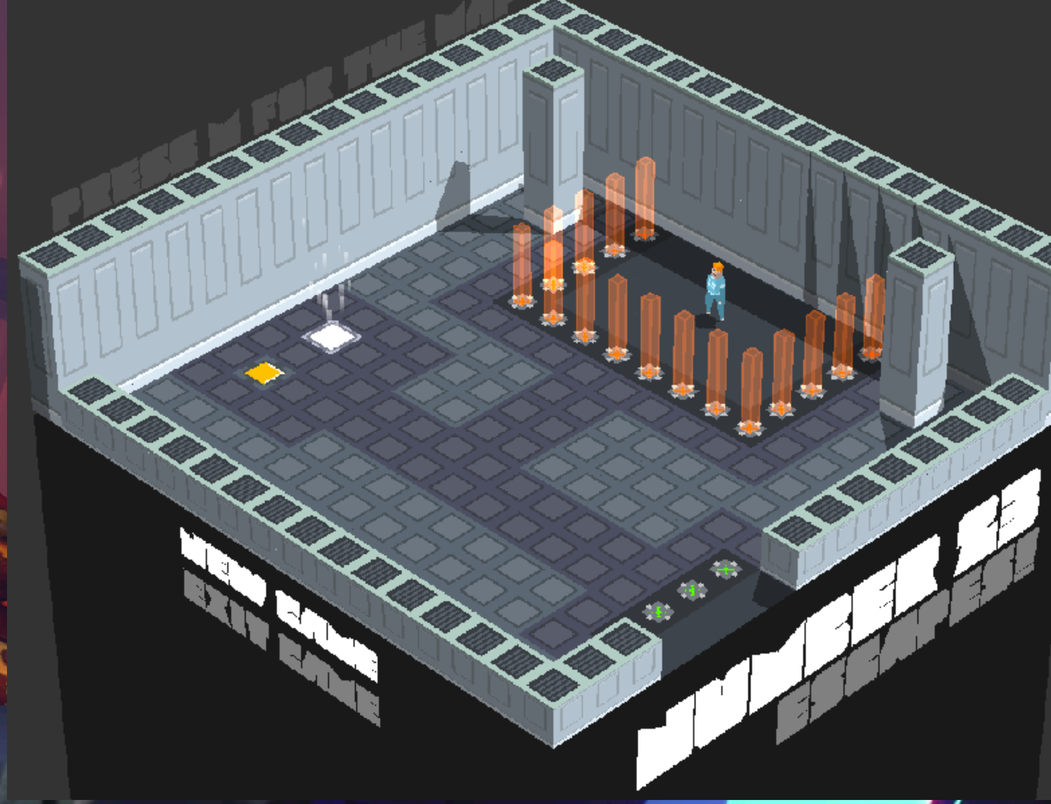
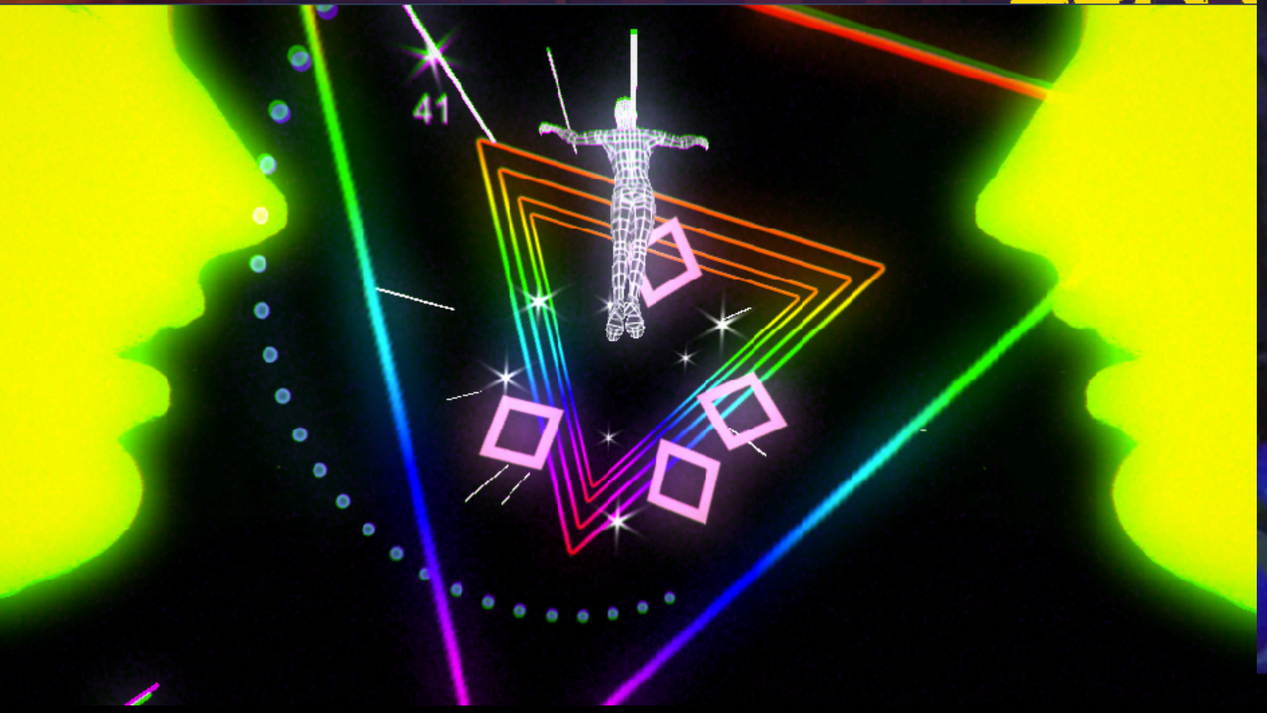
CS-3113
Introduction to game programming

Hello!

Prof. Ivan Safrin



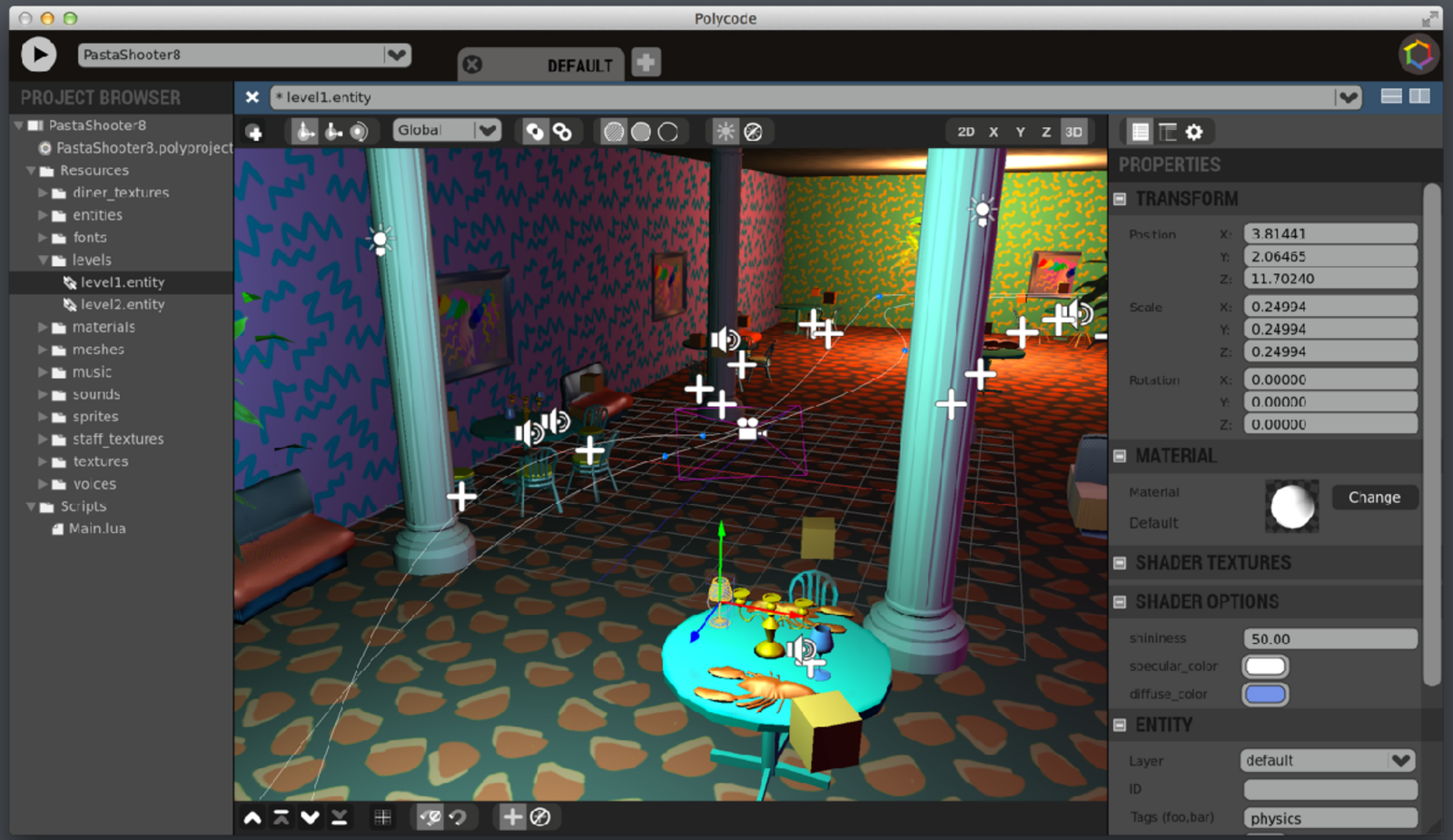



41


MORE VIDEOS


TEAM1
TEAM2
04:52

Polycode



polycode.org


IBM
TOUCH THE SCREEN TO LEARN ABOUT
IBM


Youzhny vs. Hewitt
Match Completed
3-2
13%
Longest Rally


#2-1!
#Vict ory!
6578 RETWEETS

The class

New York University Tandon School of Engineering
Department of Computer Science and Engineering

Intro to Game Programming · CS-3113 · Spring 2018

Mondays and Wednesdays 12:30 p.m. - 1:50 p.m. · Rogers Hall, Room 425

Professor Ivan Safrin
Email: is1296@nyu.edu
Office hours: Mondays and Wednesdays 2:00 p.m. - 3:00 p.m (after class)

Prerequisites

CS-2134 Data Structures and Algorithms.

Course Description

"Intro to Game Programming" is a comprehensive course designed to provide an overview of modern game programming practices and their hands-on implementations. We will study the basic building blocks of interactive computer games and learn how to implement them as cross-platform C++ applications using the SDL and OpenGL libraries. Throughout the course, we will be building simple game prototypes around the topics learned in class, building up to a final game project to be completed by the end of the semester.

Course Topics

- Cross-platform game programming using C++ and SDL.
- Rendering 2D and 3D graphics using OpenGL ES2.
- Basic game physics and collision detection.
- Matrix algebra, linear transformations.
- Animation and special effects.

Course Overview

1. Introductions, overview of tools, setting up a development environment.
2. Basic graphics using OpenGL.
3. Input, time-based movement, basic collision detection.
4. Sprites, text and sprite animation.
5. Game physics, fixed time step.
6. Level editing and procedural generation.
7. Playing sounds.
8. Matrix transformations, advanced collision detection.
9. Particle systems, timeline animation.
10. AI programming.
11. Introduction to 3D graphics.
12. Advanced 3D graphics.
13. Shaders, lighting and post-processing.
14. Using an external physics engine.
15. Cross-platform and mobile development.
16. Final project demos.

Reading materials

None required. All needed information will be provided in class and online, however the books below are recommended supplementary reading.

Class slides and other helpful materials are available on Github at
https://github.com/ivansafrin/CS3113

Very comprehensive book covering in great depth many of the concepts of graphics programming that we will be studying:
Foundations of 3D Computer Graphics (Steven J. Gortler)
http://www.amazon.com/Foundations-Computer-Graphics-Steven-Gortler/dp/0262017350/

And if you need to brush up on your C++:
Programming: Principles and Practice Using C++ (2nd Edition) (Bjarne Stroustrup)
http://www.amazon.com/dp/0321992784/

Grading / Assignments

Each week, you will have an assignment to complete by the beginning of next week's first class. Most of these assignments will involve implementing the concepts we learned that week as a simple game prototype. As part of the class, you will be expected to create a Github account if you don't already have one and submit your assignments via a git repository (we will go over how to do this during the first class). **Assignments must be submitted on time (before next week's first class, unless otherwise specified) or your assignment grade will be docked half a grade for each day it is late.**

You will also be expected to create a larger game by the end of the semester, which will serve as your final project. You may team up with another person for this project.

Your final course grade will be based on completion of weekly assignments (40%), tests (20%) and your final game project (40%).

Attendance Policy

Attendance will be taken every class. If **you miss more than three classes, your final grade will be affected.** Please keep in mind that we have a lot of material to cover in a fairly short time and missing even a single class will likely set you back!

Use of external code

For all of the assignments, including the final project, you will be expected to write all of the code yourself. Some example and helper code will be provided for you in class and online. If you wish to use other code, such as an open-source library or snippets of open-source code to implement features **NOT** covered in the class, you may do so, **but you must check in with me** beforehand and all open-source code used in your projects must be clearly marked and properly attributed.

https://github.com/ivansafrin/CS3113

Slack.

cs3113spring2018.slack.com

A word of warning!

Prerequisites

CS2134 Data Structures and Algorithms.

Surprise C++ quiz!

What are games?

What are video games?

What is game programming?

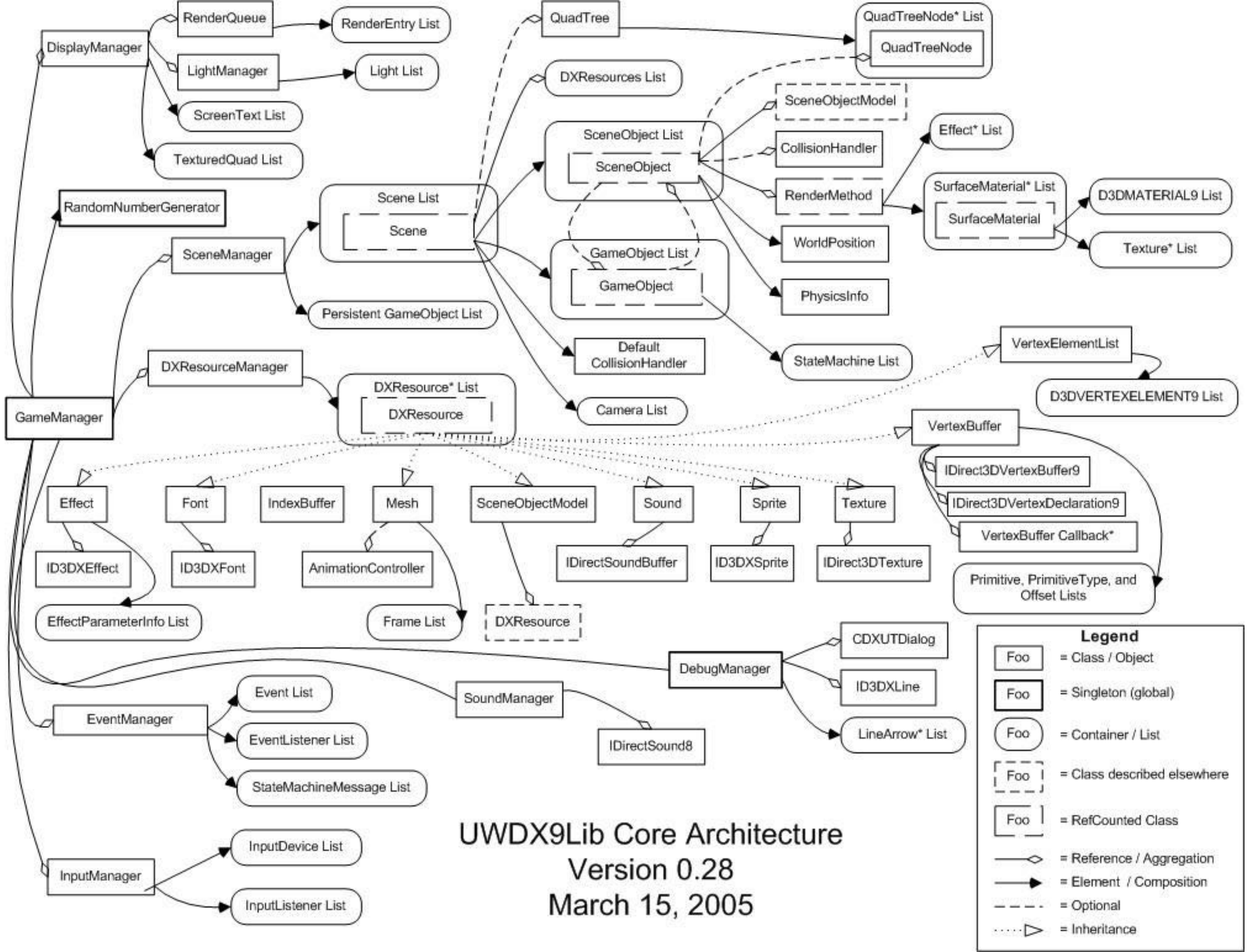
What are the common game components?

What are the common game components?

- Graphics
- Input
- Sound
- Game Logic (Physics, AI)

Typical game structure.

```
Setup();
while(gameIsRunning) {
    ProcessInput();
    UpdateGameWorld();
    Render();
}
```


RenderQueue
RenderEntry List
DisplayManager
LightManager
Light List
ScreenText List
TexturedQuad List
QuadTree
QuadTreeNode* List
QuadTreeNode
DXResources List
SceneObjectModel
SceneObject List
SceneObject
CollisionHandler
Effect* List
RenderMethod
SurfaceMaterial* List
SurfaceMaterial
D3DMATERIAL9 List
Texture* List
WorldPosition
PhysicsInfo
RandomNumberGenerator
Scene List
Scene
SceneManager
GameObject List
GameObject
Persistent GameObject List
Default CollisionHandler
StateMachine List
Camera List
VertexElementList
D3DVERTEXELEMENT9 List
DXResourceManager
DXResource* List
DXResource
GameManager
VertexBuffer
IDirect3DVertexBuffer9
IDirect3DVertexDeclaration9
VertexBuffer Callback*
Effect
Font
IndexBuffer
Mesh
SceneObjectModel
Sound
Sprite
Texture
ID3DXEffect
ID3DXFont
AnimationController
IDirectSoundBuffer
ID3DXSprite
IDirect3DTexture
Primitive, PrimitiveType, and Offset Lists
EffectParameterInfo List
Frame List
DXResource
DebugManager
CDXUTDialog
ID3DXLine
LineArrow* List
SoundManager
Event List
EventManager
EventListener List
StateMachineMessage List
IDirectSound8
InputManager
InputDevice List
InputListener List
UWDX9Lib Core Architecture
Version 0.28
March 15, 2005
Legend
Foo = Class / Object
Foo = Singleton (global)
Foo = Container / List
Foo = Class described elsewhere
Foo = RefCounted Class
= Reference / Aggregation
= Element / Composition
= Optional
= Inheritance

Our tools

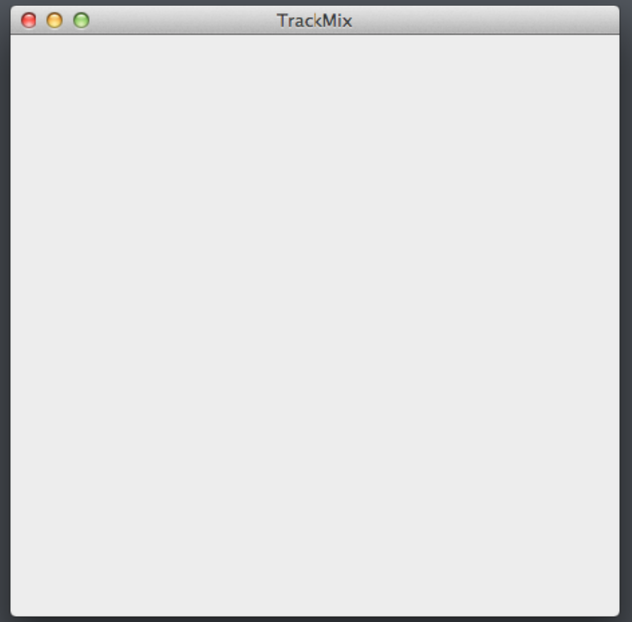
C++

Libraries:
SDL (Input, Windowing, Sound)
OpenGL (Graphics)

Why C++ and OpenGL?

- Widely used for game development
- Fast and free
- Allows us to focus on general programming concepts
- Builds a low level understanding

What is SDL?

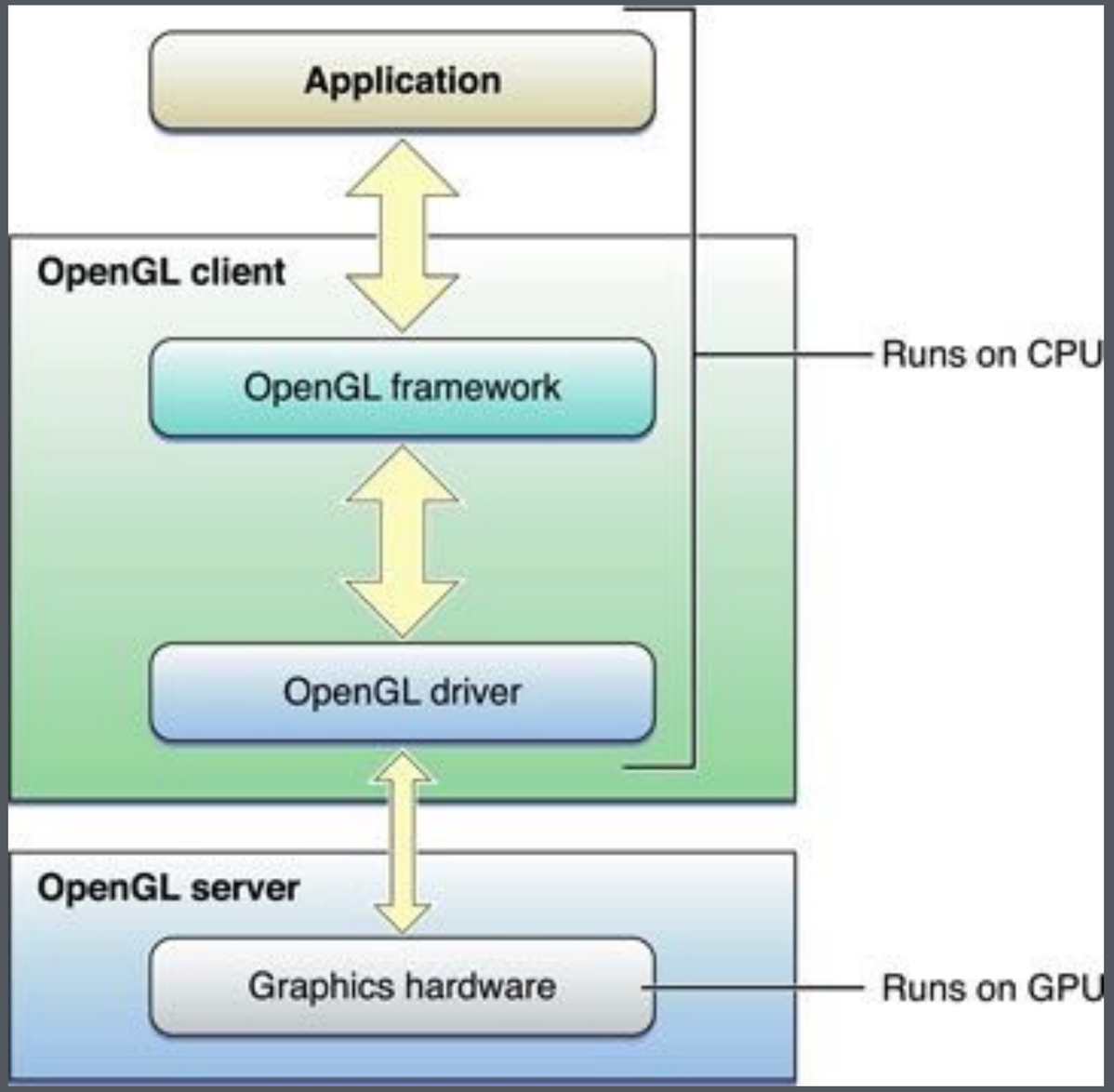

TrackMix


OpenGL


60
5
10
15
20
25
30
35
40
45
50
55


SDL
Simple Directmedia Layer
TraccMix
OpenGL

What is OpenGL?


Application
OpenGL client
OpenGL framework
OpenGL driver
Runs on CPU
OpenGL server
Graphics hardware
Runs on GPU

Setup your development environment

Setup a Github account.

github.com

Install git if you don’t already have it

git-scm.com

If you don't know how to use git,
follow this tutorial.

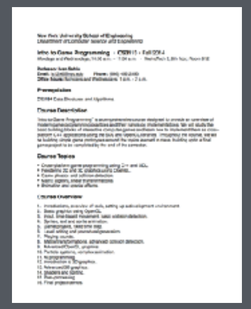
try.github.io

Clone the class repository at

github.com/ivansafrin/CS3113

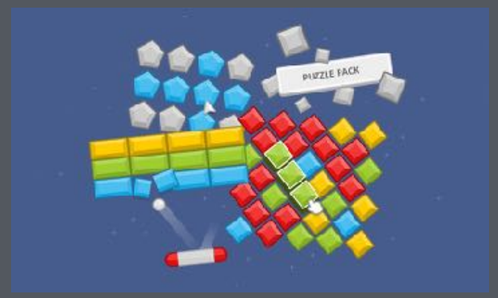
```
git clone https://github.com/ivansafrin/CS3113.git
```

What's in the repo?



Syllabus

Project templates and helper code



CC0 Game Assets

Class slides

Create a Github repository for this class

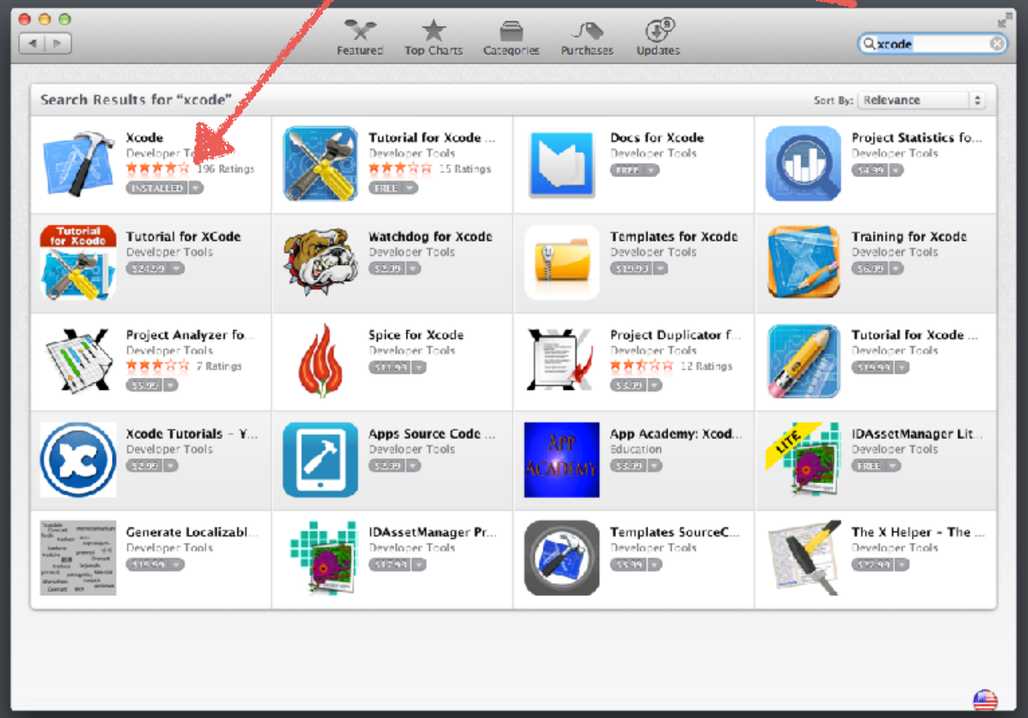
https://help.github.com/articles/create-a-repo

Do not make your repository a fork of the class repository! Start with an empty one!

Install Xcode if you are on Mac or Visual Studio if you are on Windows.

To install Xcode

Search for Xcode in the AppStore

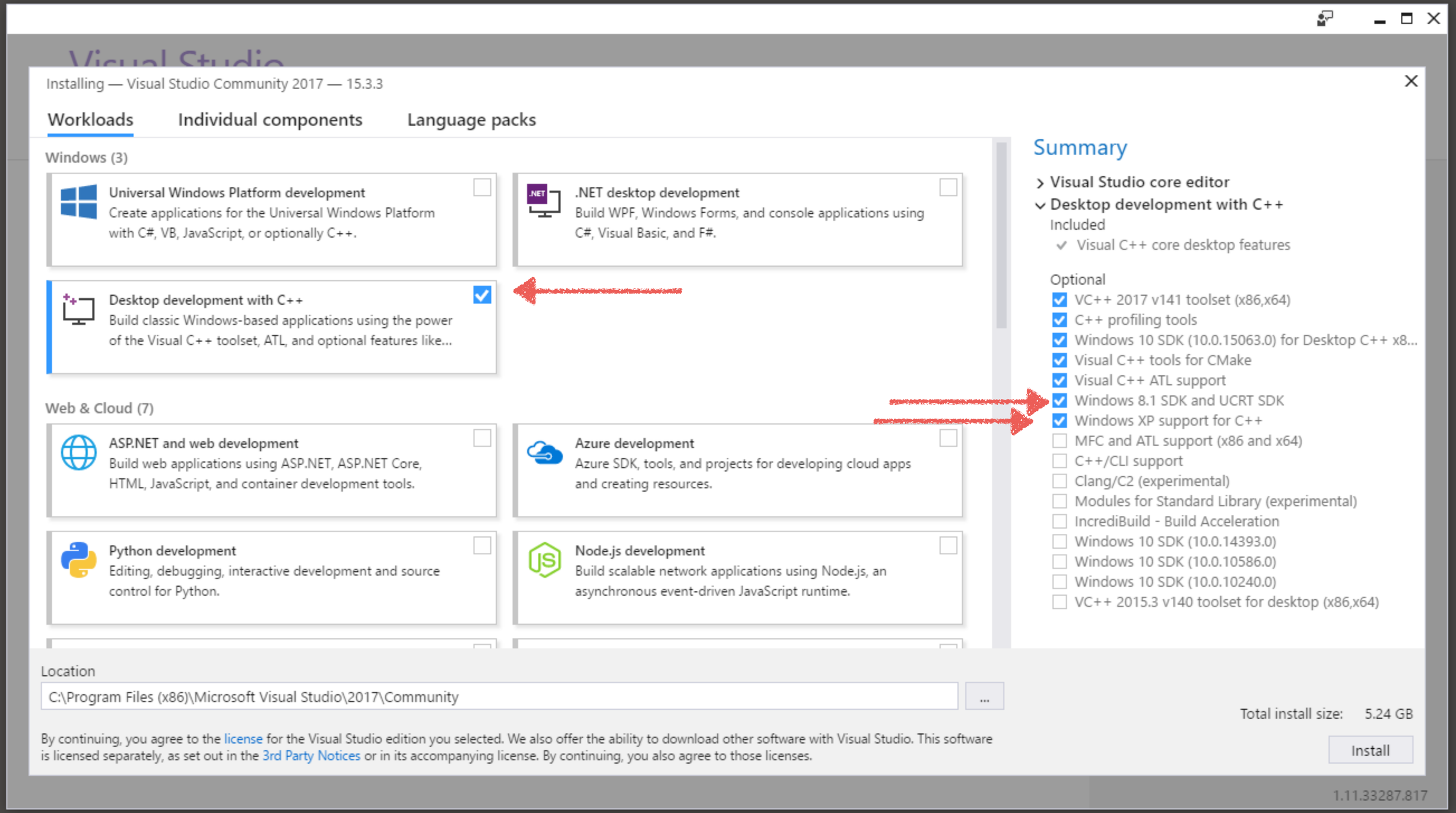


Click Install

To install Visual Studio Community

Go to

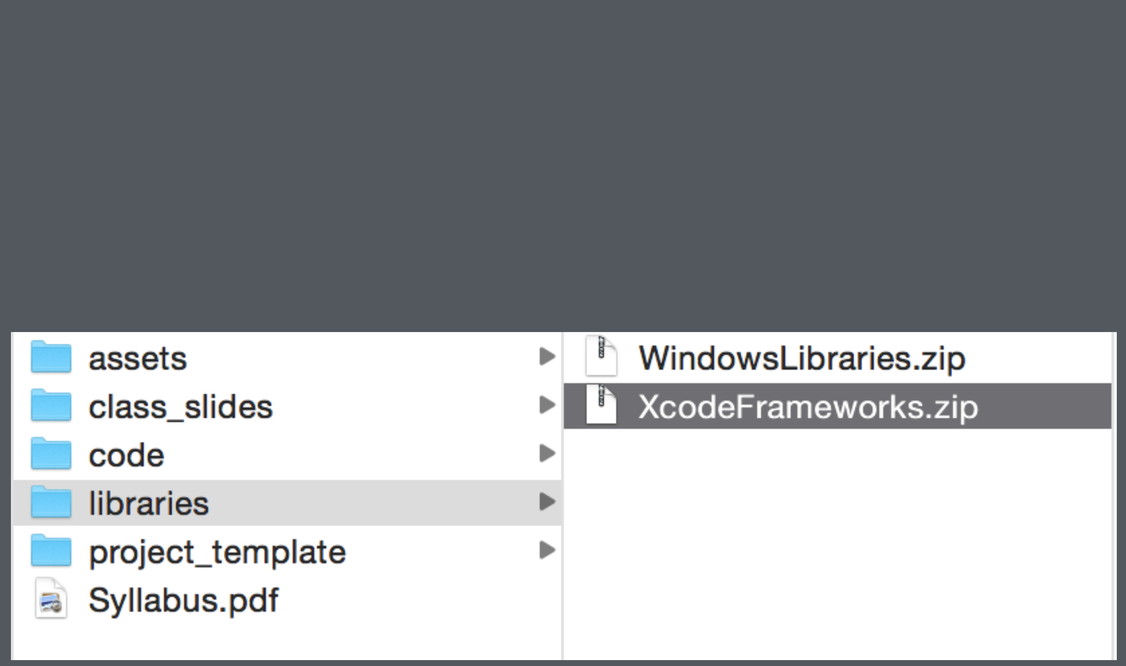
https://www.visualstudio.com/vs/visual-studio-express/

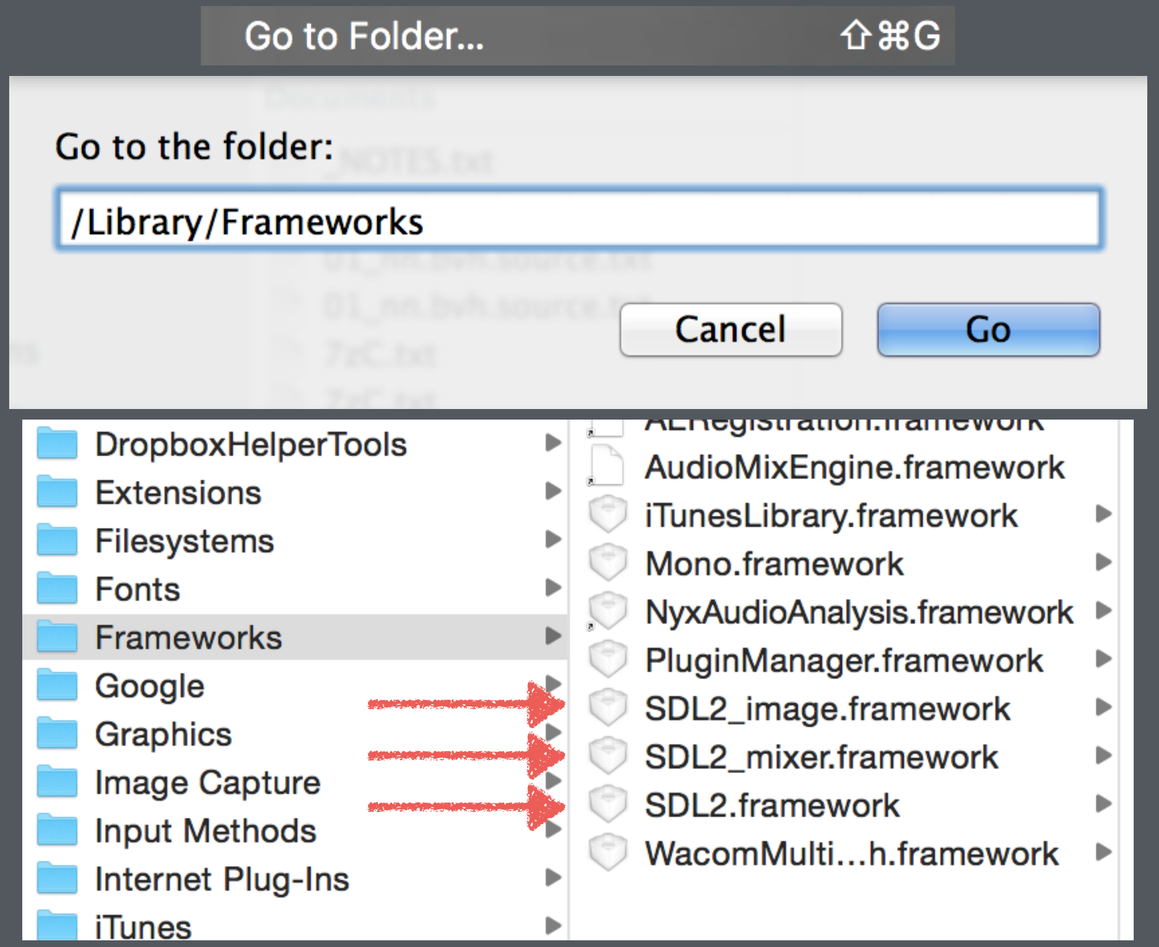

Visual Studio
Installing — Visual Studio Community 2017 — 15.3.3
Workloads
Individual components
Language packs
Windows (3)
Universal Windows Platform development
Create applications for the Universal Windows Platform with C#, VB, JavaScript, or optionally C++.
.NET desktop development
Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F#.
Desktop development with C++
Build classic Windows-based applications using the power of the Visual C++ toolset, ATL, and optional features like...
Web & Cloud (7)
ASP.NET and web development
Build web applications using ASP.NET, ASP.NET Core, HTML, JavaScript, and container development tools.
Azure development
Azure SDK, tools, and projects for developing cloud apps and creating resources.
Python development
Editing, debugging, interactive development and source control for Python.
Node.js development
Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.
Summary
Visual Studio core editor
Desktop development with C++
Included
Visual C++ core desktop features
Optional
VC++ 2017 v141 toolset (x86,x64)
C++ profiling tools
Windows 10 SDK (10.0.15063.0) for Desktop C++ x8...
Visual C++ tools for CMake
Visual C++ ATL support
Windows 8.1 SDK and UCRT SDK
Windows XP support for C++
MFC and ATL support (x86 and x64)
C++/CLI support
Clang/C2 (experimental)
Modules for Standard Library (experimental)
IncrediBuild - Build Acceleration
Windows 10 SDK (10.0.14393.0)
Windows 10 SDK (10.0.10586.0)
Windows 10 SDK (10.0.10240.0)
VC++ 2015.3 v140 toolset for desktop (x86,x64)
Location
C:\Program Files (x86)\Microsoft Visual Studio\2017\Community
...
Total install size: 5.24 GB
By continuing, you agree to the license for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the 3rd Party Notices or in its accompanying license. By continuing, you also agree to those licenses.
Install
1.11.33287.817

Install SDL libraries

On Mac

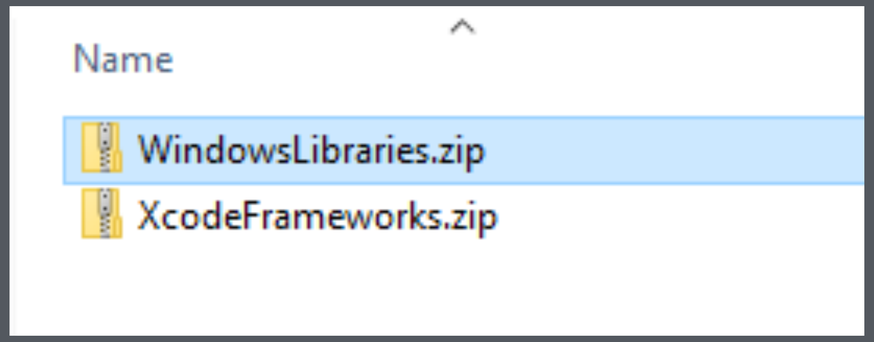
Unzip XcodeFrameworks.zip under “libraries” in the class repository to /Library/Frameworks

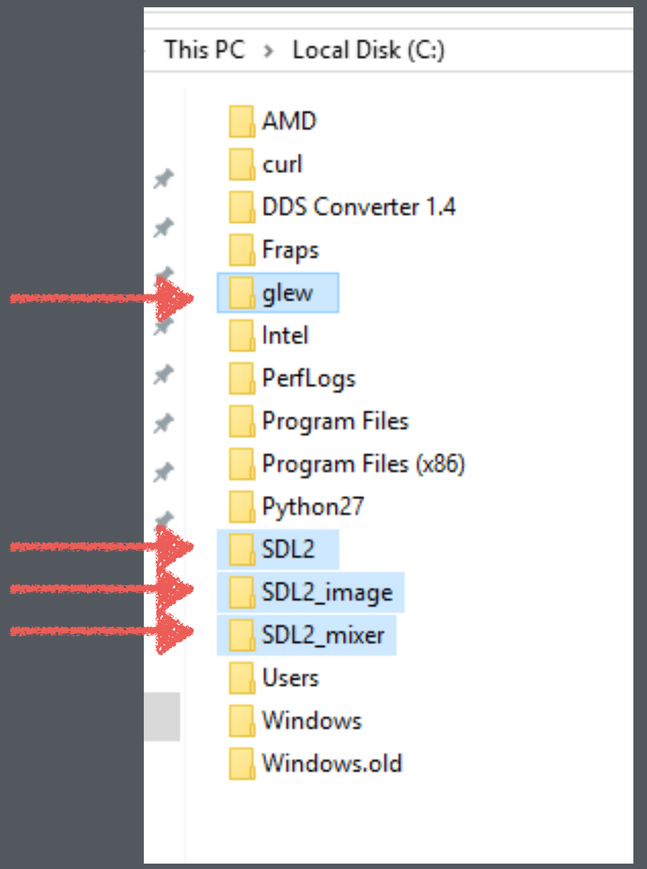




On Windows

Unzip WindowsLibraries.zip under libraries in the class repository to C:\





Build and run the template project from the class git repository.

SDL **boilerplate code**.

```
#ifdef _WINDOWS
    #include <GL/glew.h>
#endif
#include <SDL.h>
#include <SDL_opengl.h>
#include <SDL_image.h>

#ifdef _WINDOWS
    #define RESOURCE_FOLDER ""
#else
    #define RESOURCE_FOLDER "NYUCodebase.app/Contents/Resources/"
#endif

SDL_Window* displayWindow;

int main(int argc, char *argv[])
{
    SDL_Init(SDL_INIT_VIDEO);
    displayWindow = SDL_CreateWindow("My Game", SDL_WINDOWPOS_CENTERED, SDL_WINDOWPOS_CENTERED, 640, 360, SDL_WINDOW_OPENGL);
    SDL_GLContext context = SDL_GL_CreateContext(displayWindow);
    SDL_GL_MakeCurrent(displayWindow, context);
    #ifdef _WINDOWS
        glewInit();
    #endif

    SDL_Event event;
    bool done = false;
    while (!done) {
        while (SDL_PollEvent(&event)) {
            if (event.type == SDL_QUIT || event.type == SDL_WINDOWEVENT_CLOSE) {
                done = true;
            }
        }
        glClear(GL_COLOR_BUFFER_BIT);
        SDL_GL_SwapWindow(displayWindow);
    }

    SDL_Quit();
    return 0;
}
```

Typical game structure.

```
Setup();
while(gameIsRunning) {
    ProcessInput();
    UpdateGameWorld();
    Render();
}
```

Your first assignment:

- Set up a github account if you don't have one.
- Install git if not installed and learn how to use it.
- Clone the class repository.
- Setup your own (empty!) repository for this class.
- Setup Xcode or Visual Studio and install the libraries.
- Make sure the template project in the class repository builds and runs!
- Sign up for the slack channel (optional).

Please email me if you run into any issues:

is1296@nyu.edu