CS 134

Actual Programming

Game Engine Architecture
Chapter 7

~~Roll Call~~
Add Codes

Actual Programming

- I will be covering this class in both C and Java
  - SDL / GLEW for C on Windows and Linux
  - Just SDL for C on Mac
  - JOGL for Java
- I am most comfortable with C on Windows and Linux

Actual Programming

- SDL and GLEW make it easy to do the things you want to do in a video game in C
- JOGL does the same thing for Java.
  - Create a window, initialize OpenGL in it
  - Run the message pump
  - Use the latest version of OpenGL
  - Get keyboard and joystick input
  - Load and draw images (Code I will provide)

Actual Programming

- JOGL – Java OpenGL
  - Main site
    http://jogamp.org/jogl/www/
  - Docs
    http://jogamp.org/deployment/jogamp-next/javadoc/jogl/javadoc/
  - Download
    http://jogamp.org/wiki/index.php/Downloading_and_installing_JOGL
  - GL Docs
    http://www.opengl.org/registry/

Actual Programming

- SDL – Simple DirectMedia Library
  - Main site http://www.libsdl.org/
  - Docs http://wiki.libsdl.org/FrontPage
  - Download http://www.libsdl.org/download-2.0.php
- GLEW – GL Extension Wrangler
  - Main site http://glew.sourceforge.net/
  - Docs No official docs, but is very simple
  - Download http://glew.sourceforge.net/index.html
  - GL Docs http://www.opengl.org/registry/

Using JOGL

```
// package imports left out

public class JavaTemplate {
    private static boolean shouldExit;
    private static boolean kbPrevState[] = new boolean[256];
    private static boolean kbState[] = new boolean[256];

    public static void main(String[] args) {
        GLProfile gl2Profile;

        try {
            // Make sure we have a recent version of OpenGL
            gl2Profile = GLProfile.get(GLProfile.GL2);
        }
        catch (GLException ex) {
            System.out.println("OpenGL max supported version is too low.");
            System.exit(1);
            return;
        }

        // Create the window and OpenGL context.
        GLWindow window = GLWindow.create(new GLCapabilities(gl2Profile));
        window.setSize(800, 600);
        window.setTitle("Java Template");
        window.setVisible(true);
```

Using JOGL

```
window.setDefaultCloseOperation(
        WindowClosingProtocol.WindowClosingMode.DISPOSE_ON_CLOSE);
window.addKeyListener(new KeyListener() {
    @Override
    public void keyPressed(KeyEvent keyEvent) {
        if (keyEvent.isAutoRepeat()) { return; }
        kbState[keyEvent.getKeyCode()] = true;
    }

    @Override
    public void keyReleased(KeyEvent keyEvent) {
        if (keyEvent.isAutoRepeat()) { return; }
        kbState[keyEvent.getKeyCode()] = false;
    }
});

// Setup OpenGL state.
window.getContext().makeCurrent();
GL2 gl = window.getGL().getGL2();
gl.glViewport(0, 0, 800, 600);
gl.glMatrixMode(GL2.GL_PROJECTION);
gl.glOrtho(0, 800, 600, 0, 0, 100);
gl.glEnable(GL2.GL_TEXTURE_2D);
gl.glEnable(GL2.GL_BLEND);
gl.glBlendFunc(GL2.GL_SRC_ALPHA, GL2.GL_ONE_MINUS_SRC_ALPHA);
```

Using JOGL

```
// The game loop
long lastFrameNS;
long curFrameNS = System.nanoTime();
while (!shouldExit) {
    System.arraycopy(kbState, 0, kbPrevState, 0, kbState.length);
    lastFrameNS = curFrameNS;
    curFrameNS = System.nanoTime();
    long deltaTimeMS = (curFrameNS - lastFrameNS) / 1000000;

    // Actually, this runs the entire OS message pump.
    window.display();

    if (!window.isVisible()) {
        shouldExit = true;
        break;
    }

    gl.glClearColor(0, 0, 0, 1);
    gl.glClear(GL2.GL_COLOR_BUFFER_BIT);
}
```

Using JOGL

- Most of this is OpenGL framework logic and won't need to change.

- You will want to customize the following sections:
- Initialization
  - Load textures, setup initial state
- Update / Draw
  - Read keyboard input, change state
  - Draw current state

Using JOGL

```
    INITIALIZATION GOES HERE
    // The game loop
    long lastFrameNS;
    long curFrameNS = System.nanoTime();
    while (!shouldExit) {
        System.arraycopy(kbState, 0, kbPrevState, 0, kbState.length);
        lastFrameNS = curFrameNS;
        curFrameNS = System.nanoTime();
        long deltaTimeMS = (curFrameNS - lastFrameNS) / 1000000;

        // Actually, this runs the entire OS message pump.
        window.display();
        if (!window.isVisible()) {
            shouldExit = true;
            break;
        }
        UPDATE GOES HERE
        gl.glClearColor(0, 0, 0, 1);
        gl.glClear(GL2.GL_COLOR_BUFFER_BIT);
        DRAWING GOES HERE
    }

    CLEANUP GOES HERE
}
```

Using SDL and GLEW

```
/* #includes left out */

char shouldExit = 0;
unsigned char kbPrevState[SDL_NUM_SCANCODES] = {0};
const unsigned char* kbState = NULL;

int main(void)
{
    // Initialize SDL.
    SDL_Init(SDL_INIT_VIDEO);

    // Create the window and OpenGL context.
    SDL_GL_SetAttribute(SDL_GL_BUFFER_SIZE, 32);
    SDL_GL_SetAttribute(SDL_GL_DOUBLEBUFFER, 1);
    SDL_Window* window = SDL_CreateWindow(
            "SDLTemplate",
            SDL_WINDOWPOS_UNDEFINED, SDL_WINDOWPOS_UNDEFINED,
            800, 600,
            SDL_WINDOW_OPENGL);

    SDL_GL_CreateContext(window);
```

Using SDL and GLEW

```
// Make sure we have a recent version of OpenGL.
GLenum glewError = glewInit();
if (glewError != GLEW_OK) {
    fprintf(stderr, "Could not initialize glew. ErrorCode=%s\n", glewGetErrorString(glewError));
    SDL_Quit();
    return 1;
}
if (GLEW_VERSION_2_0) {
    fprintf(stderr, "OpenGL 2.0 or greater supported: Version=%s\n",
            glGetString(GL_VERSION));
} else {
    fprintf(stderr, "OpenGL max supported version is too low.\n");
    SDL_Quit();
    return 1;
}

// Setup OpenGL state.
glViewport(0, 0, 800, 600);
glMatrixMode(GL_PROJECTION);
glOrtho(0, 800, 600, 0, 0, 100);
glEnable(GL_TEXTURE_2D);
glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
```

Using SDL and GLEW

```
    // The game loop.
    kbState = SDL_GetKeyboardState(NULL);
    while (!shouldExit) {
        assert(glGetError() == GL_NO_ERROR);
        memcpy(kbPrevState, kbState, sizeof(kbPrevState));

        // Handle OS message pump.
        SDL_Event event;
        while (SDL_PollEvent(&event)) {
            switch (event.type) {
                case SDL_QUIT:
                    shouldExit = 1;
            }
        }

        glClearColor(0, 0, 0, 1);
        glClear(GL_COLOR_BUFFER_BIT);

        // Present the most recent frame.
        SDL_GL_SwapWindow(window);
    }

    SDL_Quit();

    return 0;
}
```

Using SDL and GLEW

```
INITIALIZATION GOES HERE
// The game loop.
kbState = SDL_GetKeyboardState(NULL);
while (!shouldExit) {
    assert(glGetError() == GL_NO_ERROR);
    memcpy(kbPrevState, kbState, sizeof(kbPrevState));

    // Handle OS message pump.
    SDL_Event event;
    while (SDL_PollEvent(&event)) {
        switch (event.type) {
            case SDL_QUIT:
                shouldExit = 1;
        }
    }
    UPDATE GOES HERE
    glClearColor(0, 0, 0, 1);
    glClear(GL_COLOR_BUFFER_BIT);
    DRAWING GOES HERE
    // Present the most recent frame.
    SDL_GL_SwapWindow(window);
}
CLEANUP GOES HERE
SDL_Quit();

return 0;
```

Handling Input

- In games, you want to poll keyboard state every frame instead of listening for events.
  - Scancodes refer to key POSITIONS, not the characters on them.
- The samples already are providing kbState and kbPrevState which you can index into by scancode
  - KeyEvent.VK_* in Java
  - SDL_SCANCODE_* in C

Handling Input

- What is different in the feel about these two games' jumping?
  - http://armorgames.com/play/2893/achievement-unlocked
  - http://www.allsonicgames.net/ultimate-flash-sonic.php
- You need to be able to detect when a key was JUST pressed or JUST released.

Handling Input

- Solution:
  - Keep the last frame's keyboard state around.
  - A key press is...
  - A key release is...

Loading / Drawing Images

- This is NOT an OpenGL class
- I am providing two functions for you:
  - glTexImageTGAFile()
  - glDrawSprite()
- These functions load a .tga file and draw it on screen.

Loading / Drawing Images

```
// Art to draw
int spriteTex;
int spriteTexSize[] = new int[2];

...

    // Load the textures.
    spriteTex = glTexImageTGAFile(gl, ”magikarp.tga”, spriteTexSize);

    // The game loop.
    while (!shouldExit ) {
        …

        gl.glClearColor(0, 0, 0, 1);
        gl.glClear(GL_COLOR_BUFFER_BIT);

        glDrawSprite(gl, spriteTex, 0, 0, spriteTexSize[0], spriteTexSize[1]);
    }

...
```

Actual Programming

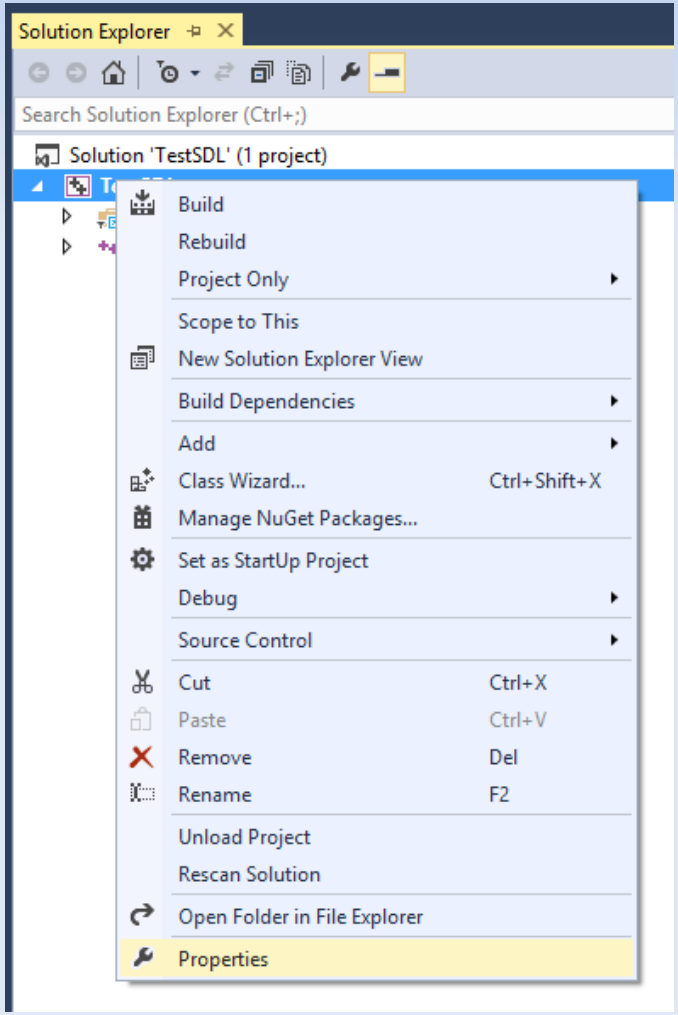
Questions?

Building Your Game

- Using external library requires three things in C
  - Compiling – Include Paths
  - Linking – Shared Library defs
  - Running – Loading Shared Libs
- Two things in Java
  - Compilng – JAR paths
  - Running – Loading JARs
- This is EXTREMELY platform specific (even with Java)

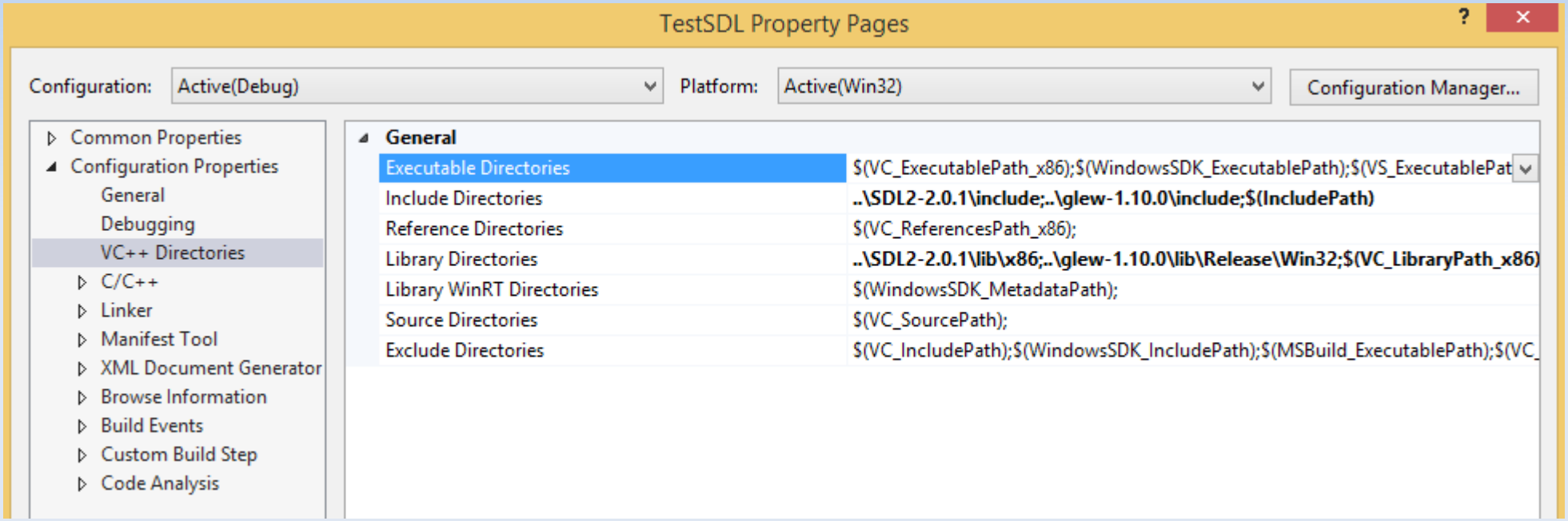
Building Your Game (Windows)

- All Settings in Visual Studio are done in a GUI



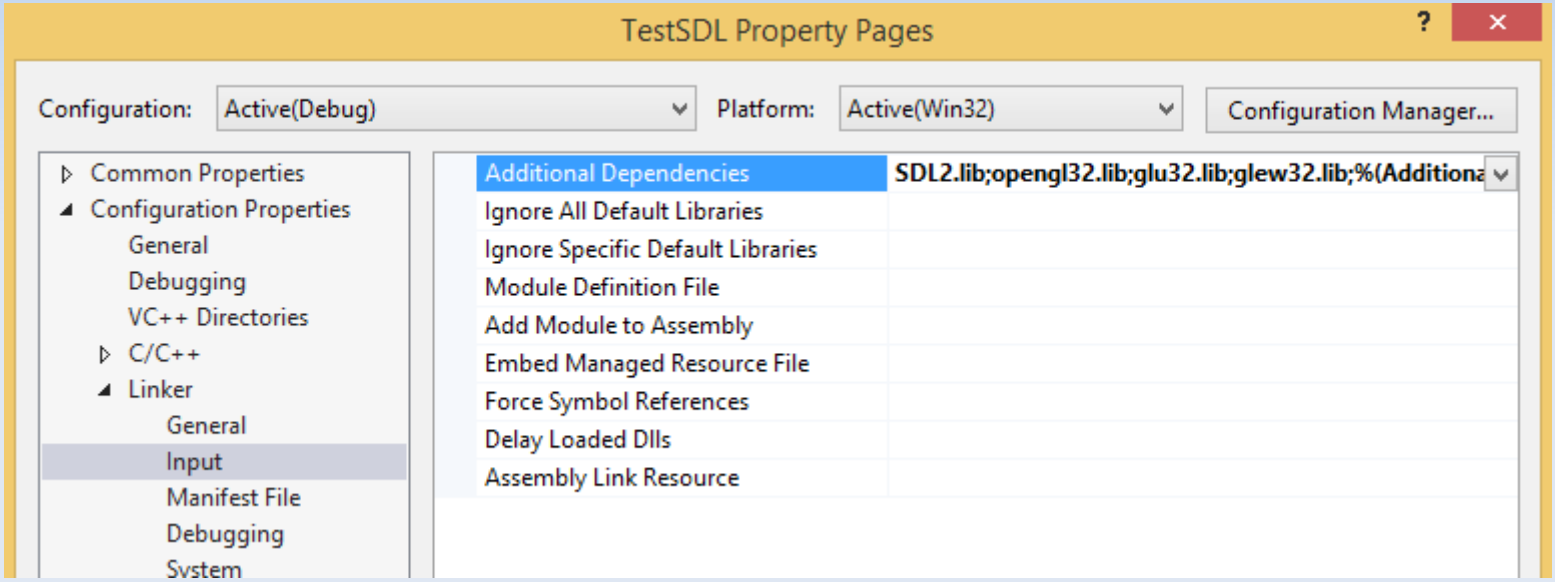
Building Your Game (Windows)

- Include Paths, Shared Library def Paths



Building Your Game (Windows)

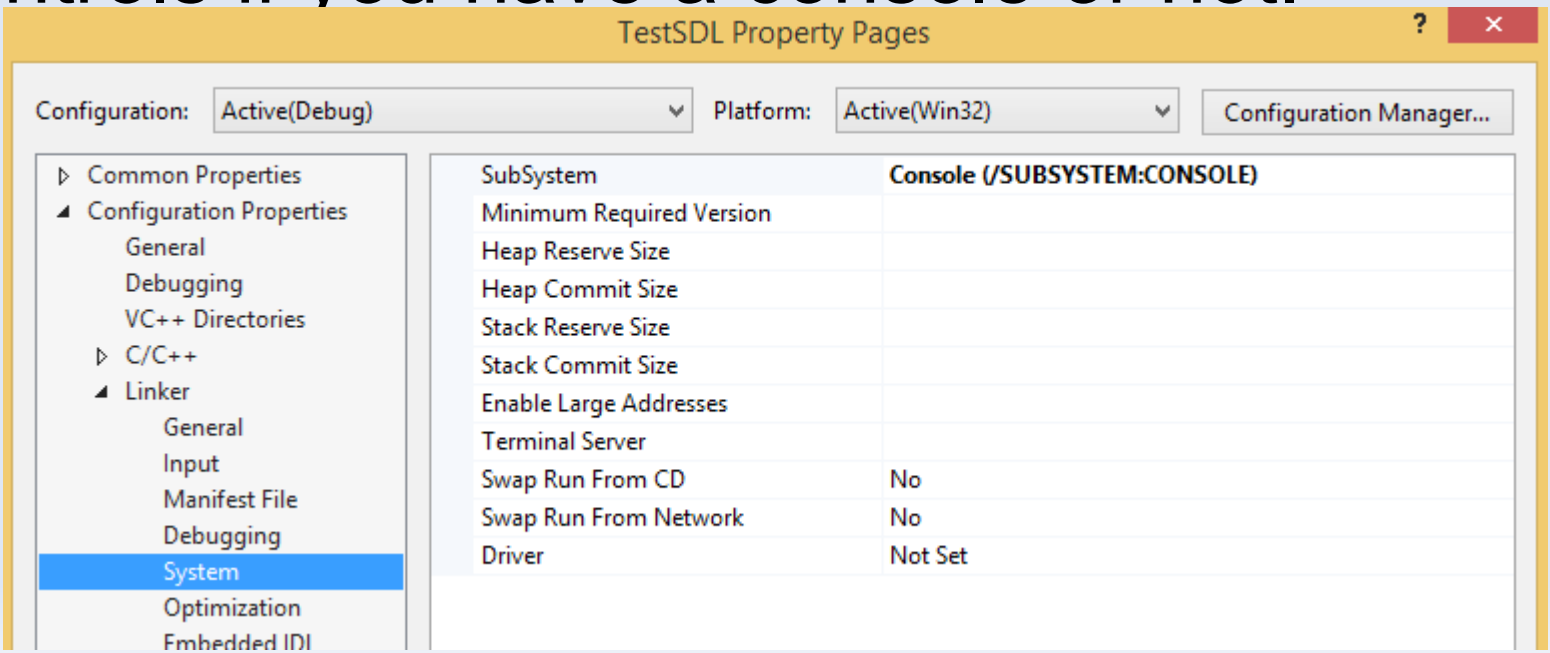
- Shared Library defs (.lib files)



Building Your Game (Windows)

- Make sure to copy the .dll files to the same folder as the .vcxproj!

- You should also know about subsystem, which controls if you have a console or not.



Building Your Game (Linux)

- Everything is set using a command line tool pkg-config

- Add `pkg-config --cflags sdl2 gl glew` compiler options (can use CFLAGS)
- Add `pkg-config --libs sdl2 gl glew` to linker options
- All in one example:
  - gcc TestSDL.cpp `pkg-config --cflags --libs sdl2 gl glew` -o TestSDL -lstdc++

Building Your Game

Questions?

Homework

- Homework is due at 11:59pm on the specified day.
- Homework is a **significant** amount of your grade, so make sure to submit every homework!

- Submit everything via Canvas
  - Source code
  - Something I can double click (.exe, .app, or .jar)

Homework

- Create a window, the window should have an image drawn in it. When a key is pressed, the image should move around.
- The image should not be able to leave the window.
- Extra credit: Have another thing that chases the ”player”.
- Hints:
  - Use glTexImageTGAFile() to load your images before the game loop.
  - Use glDrawSprite() to draw an image – you should remember how big the image is.