CSCI 200: Foundational Programming Concepts & Design Lecture 24

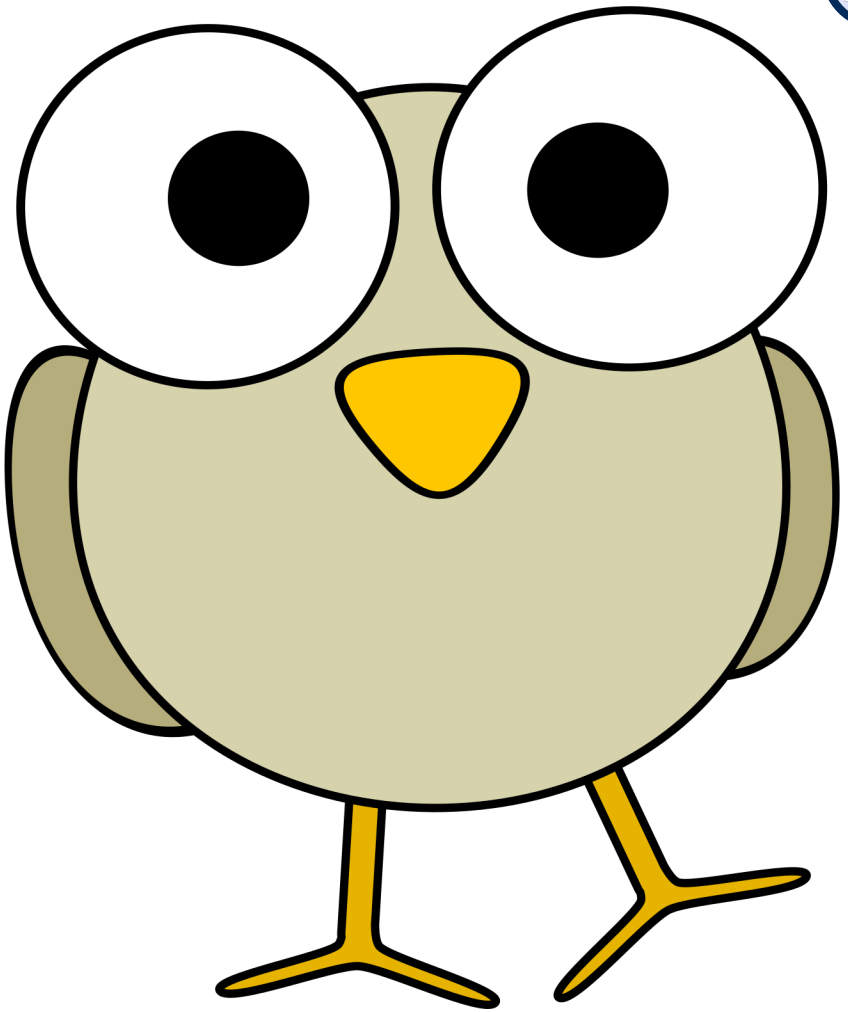
Making & Using Libraries

Download SFML Template for **lab machines** to follow along

Previously in CSCI 200

- Templates
  - Abstract data type
  - Cannot precompile
  - Use hpp

Questions?



Learning Outcomes For Today

- Explain what a library archive is
- Discuss reasons why headers & implementations can be bundled into a library archive and distributed
- Build and install a third-party library (SFML) to use within a C++ program (to display graphics)

On Tap For Today

- Using Libraries
- The SFML Library
- Practice

On Tap For Today

- Using Libraries
- The SFML Library
- Practice

Warehouse

```
#include "Box.h"
#include <vector>

class Warehouse {
public:
   ...
private:
   std::vector<Box*> *_pBoxen;
   ...
};
```

Warehouse

```
#include "Box.h"
#include <vector>

class Warehouse {
public:
   ...
private:
   std::vector<Box*> *_pBoxen;
   ...
};
```

- Where is **`Box.h`** header file located?

Warehouse

```
#include "Box.h"
#include <vector>

class Warehouse {
public:
   ...
private:
   std::vector<Box*> *_pBoxen;
   ...
};
```

- Where is **`Box.h`** header file located?
  - Alongside **`Warehouse.h`**

Warehouse

```
#include "Box.h"
#include <vector>

class Warehouse {
public:
   ...
private:
   std::vector<Box*> *_pBoxen;
   ...
};
```

- Where is **`vector`** header file located?

Including Headers

```
#include "LocalHeader.h"

#include <SystemHeader.h>
```

Finding System Headers

- Windows: Inside MinGW hierarchy
- OS X: Inside XCode hierarchy

```
jpaone@Havenwood-2 ~ % ls /Library/Developer/CommandLineTools/usr/include/c++/v1
__bit_reference                 fenv.h
__bsd_locale_defaults.h         filesystem
__bsd_locale_fallbacks.h        float.h
__config                        forward_list
__cxxabi_config.h               fstream
__debug                         functional
__errc                          future
__functional_03                 initializer_list
__functional_base               inttypes.h
__functional_base_03            iomanip
__hash_table                    ios
__libcpp_version                iosfwd
__locale                        iostream
__mutex_base                    istream
__node_handle                   iterator
__nullptr                       latch
__split_buffer                  limits
__sso_allocator                 limits.h
__std_stream                    list
__string                        locale
__threading_support             locale.h
__tree                          map
```

Warehouse

```
#include "Box.h"
#include <vector>

class Warehouse {
public:
   ...
private:
   std::vector<Box*> *_pBoxen;
   ...
};
```

- Where is **`vector`** header file located?
- Where is **`vector`** implementation located?

Finding System Libraries

- Again, from MinGW / XCode
- libstdc++ contains the C++ Standard Libraries

```
jpaone@Havenwood-2 ~ % ls /usr/lib
charset.alias                   libstdc++.6.dylib
cron                            log
dsc_extractor.bundle            pam
dtrace                          pkgconfig
dyld                            python2.7
groff                           rpcsvc
libLeaksAtExit.dylib            ruby
libMTLCapture.dylib             sasl2
libffi-trampolines.dylib        sqlite3
libgmalloc.dylib                ssh-keychain.dylib
libhunspell-1.2.0.dylib         swift
libiodbc.2.dylib                system
libiodbcinst.2.dylib            updaters
libobjc-trampolines.dylib       xpc
libpython.dylib                 zsh
libpython2.7.dylib
```

Archive File

- What's a library archive file?
  - Collection of object files
- What's an object file?
  - Compiled binary representation of C++ source code

Why Use Archive Files As Libraries?

- Single source of truth / Reuse
  - Headers & Implementations live in one place for ALL programs to access & use
- Distribution
  - Share a stable version of classes/functions for others to use

Distribution

1. Share header files
   - Serves as form of documentation for interface that is available
   - Necessary for other programs to include to be able to compile with your declared components

2. Share precompiled object files
   - Implementation doesn't change (compile once, use many)
   - Abstract (& hide) details of implementation (may contain proprietary algorithm)

Header Best Practices

- Headers should be minimal, only include what is absolutely necessary for interface
- Don't include any headers in your header that aren't required in that file
  - Anything that can go into source file should
- Don't use **`using namespace`** which forces others to as well

Informing the Compiler

- Tell the compiler where to look for header files
- Compiler command template

```
$(CXX) $(CFLAGS) -o $(OBJ_NAME) -c $(SRC_NAME) -I$(INC_PATH)
```

- For instance

```
g++ -Wall -g -std=c++17 -o main.o -c main.cpp -IZ:\CSCI200\include
```

- Specify location in Makefile

Informing the Linker

- Tell the linker where to look for library files
- Linker command template

```
$(CXX) -o $(TARGET_NAME) $(OBJECT_NAMES) -L$(LIB_PATH) $(LIBS)
```

- For instance

```
g++ -o SFMLExample.exe main.o -LZ:\CSCI200\lib -lsfml-system
```

- Specify location and libraries in Makefile

Makefile Updates

```
# customize your build
TARGET = TestLibrary
SRC_FILES = main.cpp
CXX = g++
CXXFLAGS = -Wall -Wextra -Werror -pedantic-errors -g -std=c++17
INC_PATH = Z:\CSCI200\include # location where headers are
LIB_PATH = Z:\CSCI200\lib     # location where archive files are
LIBS = -lTestLib              # name of archive file to load
# do not edit below here
OBJECTS = ${SRC_FILES:.cpp=.o}
DEL = ...
all: ${TARGET}
${TARGET}: ${OBJECTS}
    ${CXX} -o $@ $^ -L${LIB_PATH} ${LIBS}
.cpp.o:
    ${CXX} ${CXXFLAGS} -o $@ -c $< -I${INC_PATH}
clean:
    ${DEL} ${TARGET} ${OBJECTS}
```

On Tap For Today

- Using Libraries
- The SFML Library
- Practice

SFML

- **S**imple & **F**ast **M**ultimedia **L**ibrary

Multimedia

Multiplatform

Multilanguage

What is SFML?

- http://www.sfml-dev.org
- Multimedia = Graphics & Audio
- Used for:
  - Games
  - Data Visualization
  - Networking
  - And much much more!

SFML Libraries

- SFML source code download consists of:
  - `SFML-2.6.0/`
    - `include/`
    - `src/`
    - *`Bunch of other stuff`*
- Need to compile the `src/` files into a library (this is the first part of L4C)
- Then
  - Need to copy the header files from `include/`
  - Need to copy the library files

SFML Header

- Include like any other library file

```
#include <SFML/Graphics.hpp>
using namespace sf;
```

- Provides functions and complex data types that are used to display graphics

- (Other SFML headers exist as needed)

Sample SFML Program

```
01 #include <SFML/Graphics.hpp>
02 using namespace sf;
03 int main() {
04   // add File I/O commands here so they occur once!
05   const int WIN_WIDTH = 640, WIN_HEIGHT = 640;
06   RenderWindow window( VideoMode(WIN_WIDTH, WIN_HEIGHT),
                          "Window Title" ); // create & open a window
07   while( window.isOpen() ) {
08     window.clear();  // clear the existing contents of the window
09     // add drawing commands here so they draw every frame
10     window.display();// display the window on screen
11     Event event;
12     while( window.pollEvent(event) ) { // check for user interaction
13       if( event.type == Event::Closed ) { // user press window X
14         window.close(); // close the window
15       }
16     }
17   }
18   return 0;
19 }
```

Drawing Shapes

- Circle

```
CircleShape circ;
circ.setRadius( 20 );
circ.setPosition( 45, 90 );
circ.setFillColor( Color::Yellow );
window.draw( circ );
```

- Rectangle

```
RectangleShape rect;
rect.setSize( Vector2f( 150, 75 ) );
rect.setPosition( 115, 120 );
rect.setFillColor( Color::Blue );
window.draw( rect );
```

Adding Color

- Use the **`setFillColor()`** function
- Use the **`setOutlineColor()`** and **`setOutlineThickness()`** functions

- Several options:
  - Built-in values:
    `Color::Red`, **`Color::Green`**, etc.
  - Custom value:
    `Color( red, green, blue )`

Drawing Text: Part I

- First, we need to load a font to use
- And make sure it loaded properly

```
Font myFont;
if( !myFont.loadFromFile( "data/arial.ttf" ) ) {
  cerr << "Could not load font" << endl;
  return -1;
}
```

Drawing Text: Part II

- Now set up our Text object

```
Text myLabel;
```

- Make use of text functions to set properties

```
myLabel.setFont( myFont );

myLabel.setString( "Hello World!" );

myLabel.setFillColor( Color::Black );

myLabel.setPosition( 400, 150 );
```

- And tell window to draw it

```
window.draw( myLabel );
```

Displaying a Picture: Part I

- First, load the image into memory
  - Like text, we needed to load the font into memory

```
Texture myTexture;
if( !myTexture.loadFromFile( "data/bubble.png" ) ) {
    cerr << "Could not load image" << endl;
    return -2;
}
```

Displaying a Picture: Part II

- Now add it to a Sprite and draw!
  - Sprite is like RectangleShape with a picture
  - Ta Da!

```
Sprite mySprite;
mySprite.setTexture( myTexture );
mySprite.setPosition( 200, 250 );
mySprite.setScale( 0.3, 0.3 );
mySprite.setColor( Color::Green );

window.draw( mySprite );
```

Animation: Part I

- Need to store position as a variable

```
double spriteX = 0, spriteY = 0;

while( window.isOpen() ) {
  // in draw loop
  mySprite.setPosition( spriteX, spriteY );
  window.draw( mySprite );
}
```

Animation: Part II

- Each iteration of draw loop = one frame

```
#include <SFML/System/Clock.hpp>

// before draw loop

double spriteX = 0, spriteY = 0;

Clock programClock;
Time lastTime = programClock.getElapsedTime();

while( window.isOpen() ) {

  // in draw loop
  mySprite.setPosition( spriteX, spriteY );
  window.draw( mySprite );

  Time currTime = programClock.getElapsedTime();
  if( (currTime - lastTime).asMilliseconds() > THRESHOLD ) {
    spriteX += dx;
    spriteY += dy;
    lastTime = currTime;
  }

}
```

Interaction = Event Handling

- Ways user can interact
  - Via keyboard
    - Key press
    - Key release
  - Via mouse
    - Button press
    - Button release
    - Mouse movement
  - And others (window minimize, lose focus, etc.)

Keyboard Interaction

- Can have user close window via keyboard

```
// in draw loop
Event event;
while( window.pollEvent(event) ) {
  if( event.type == Event::KeyPressed ) {
    switch( event.key.code ) {
    case Keyboard::Q:
      window.close();
      break;
    }
  }
}
```

Mouse Clicks

- Have sprite jump to mouse click location

```
if( event.type == Event::MouseButtonPressed ) {
    spriteX = event.mouseButton.x;
    spriteY = event.mouseButton.y;
}
```

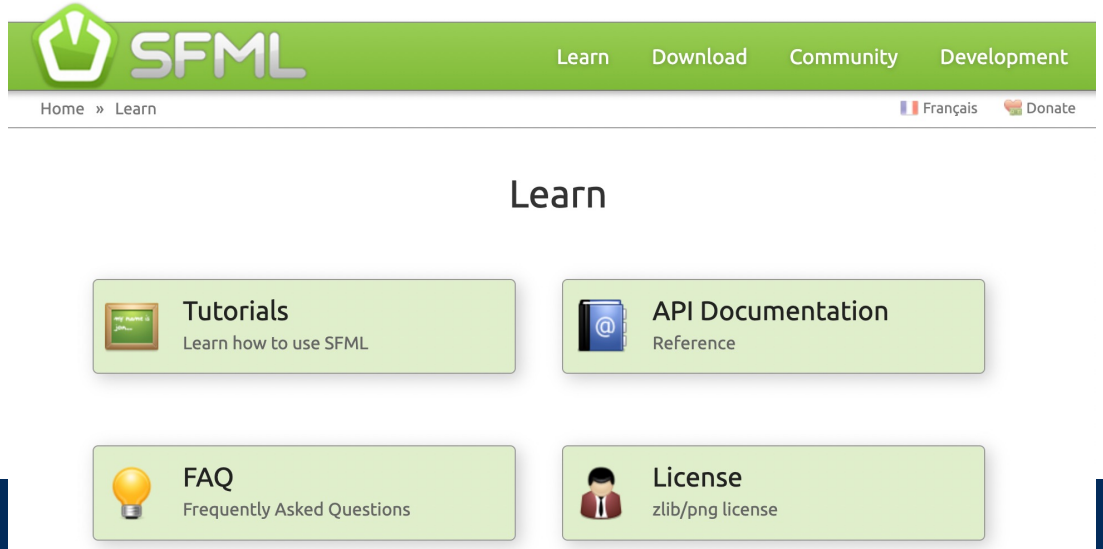
Event Handling

- Check all event types

```
Event event;
while( window.pollEvent(event) ) {
  if( event.type == Event::Closed ) {
    // close window
  } else if( event.type == Event::KeyPressed ) {
    // do something based on which key is pressed
  } else if( event.type == Event::MouseButtonPressed ) {
    // do something based on mouse click location/button
  }
}
```

SFML Documentation

- Tutorials: https://www.sfml-dev.org/tutorials/2.6/
- API Documentation: https://www.sfml-dev.org/documentation/2.6.0/
- FAQ: https://www.sfml-dev.org/faq.php



Building Against SFML – Makefile

```
# if not placed in system folders
# location where SFML headers are
INC_PATH = Z:\CSCI200\include
# location where SFML archive files are
LIB_PATH = Z:\CSCI200\lib

# name of archive files to load
LIBS = -lsfml-window -lsfml-graphics -lsfml-system -lsfml-audio -lsfml-network
```

On Tap For Today

- Using Libraries
- The SFML Library
- Practice

To Do For Next Time

- L4C
  - Build and install SFML on your machine
  - OS dependent but cross-platform
  - YMMV – ask for help on Ed!
- A4
  - Draw balls bouncing around the screen
  - Add/remove balls
- Ask for help on Ed about setting up SFML!