Lecture 14
Final Project Overview

CS211 – Fundamentals of Computer Programming II

Branden Ghena – Spring 2023

Slides adapted from:
Jesse Tov

Northwestern

Administrivia

- EX6 due today
  - Last one, hopefully shouldn’t take too long
  - Intentionally picked stuff that’s good prep for Homework 5

- Homework 5 due Thursday
  - This is the last homework too!
  - **Definitely the hardest. Be careful here!**
    - View might be more work than you expect
  - Reminder: no slip days on the final project

Administrivia

- Quiz today! (we’ll stop at 3pm to take it)

- Final project starting!
    - Proposals are due Friday
    - More details right now!

- Reminder: project partners
    - You’re allowed to work with a partner if you want
    - If you don’t know anyone, fill out the form and we’ll match you (closes tomorrow)

Today’s Goals

- Explain the what, why, and how of final projects

- Explore GE211 functionality not used in the homeworks

- Demonstrate some additional games you’ll get as sample code

- Practice the creation of a GE211 game

Getting the code for today

- Download code in a zip files from here:
  https://nu-cs211.github.io/cs211-files/hw/project_demos.zip
  https://nu-cs211.github.io/cs211-files/hw/ge211_examples.zip
  https://nu-cs211.github.io/cs211-files/hw/final_project.zip

- Extract code wherever

- Open with CLion
  - Make sure you open the folder with the CMakeLists.txt

Outline

- **Final Project Overview**
- Demo Games
- Additional GE211 Functionality
- Example: “snake game”

Goals of the Final Project

- Focus on something that interests you
  - Pick anything you like (that’s the right difficulty)
  - Chance to apply creativity and make something fun

- Program without safety rails or constraints
  - Starter code is very minimal
  - No specification with required functions to implement

  - You get to design how the code works
  - You can base your design off examples though!

Timeline

- https://nu-cs211.github.io/cs211-files/hw/final_project.pdf

- Friday, May 19 - Proposal
  - This week! (but only requires a one-sentence proposal)

- Tuesday, May 23 - Specifications
  - Next week

- Friday, June 02 - Code due
  - Last Friday of classes
  - Two full weeks to work on it

- Sunday, June 04 - Evaluation guide
  - We'll grade them during exam week, and you can focus on other stuff

Making proposals

- Something that interests you
  - Games are most common
  - I’ll let you know if it’s too easy or too complicated

- Good sources of inspiration
  - Classic arcade games
  - 2D mobile games
  - Board games

- Common problematic submissions
  - Pong, Snake game, Space Invaders, Flappy Bird
  - Any of the demo games: Keyracer, Bejewled, Asteroids

Making specifications

- List of 10-12 functionalities that your project will have
  - This is where difficulty is *really* decided
  - Grade is determined by whether you meet the specifications you create

- This is an iterative process
  - Submit spec items
  - Hear back from shepherd about what’s good and bad
  - Make updates and repeat

  - Goal:
    - Difficult enough to help you learn
    - Easy enough to complete

How to get started

1. Start with the model
   - Make the simplest version of the game that can do _anything_

2. Then implement a View and Controller so you can play it
   - Again, focus on the simple parts first

3. Then go back to model and add features

4. Finally, go back to View and Controller and add features
   - Sound, Better Graphics, etc.

Remember that simpler is often better

- If you're making a board game, you could take all of board.cxx and board.hxx and reuse it in your project
    - But it's complicated and you'll have to adjust some things for your game which will require understanding the code
    - Likely not the simplest path

- Alternative options
    - `std::vector<Posn>` track board locations for each player
    - `std::unordered_map<Posn, Player>` mark player for each location

Outline

- Final Project Overview
- **Demo Games**
- Additional GE211 Functionality
- Example: “snake game”

A note on these demos

- All kinds of complicated C++ stuff going on here
  - Some of it is good
  - Some of it is just messy

- Purpose of the demo code is to inspire you about what’s possible

- Not recommended to use one of these as a starting point
  - Too much stuff going on that wouldn’t be relevant

Getting demo code

- https://nu-cs211.github.io/cs211-files/hw/project_demos.zip

- Includes three separate projects
  - Keyracer
  - Bejeweled
  - Asteroids

Keyracer

- Practice typing words under time pressure

- Loads information from a Resource file containing all English words
    - `load_dictionary()` in `controller.cxx`
    - As you’ll see, this dictionary is a bit dubious…

- Timer bar counts down until you’ve “missed” the letter
    - Also miss if you hit the wrong key

    - Counts down time in `on_frame()`
    - Uses a Transform to scale the timer bar

Bejeweled

- Align groups of colored circles in a grid to score them
  - Makes the group disappear, scoring points
  - More colored circles fall down from the top

- Uses background music (optionally) and sound effects
  - Sound effects play when scoring or when an invalid move is made

- “Animates” steps when scoring
  - Circles disappear from the screen over several frames
  - Then circles fall down from top over several frames

Asteroids

- Avoid or shoot asteroids in a spaceship that has momentum
  - Asteroids that are shot break into multiple smaller pieces
  - Ship gains or loses velocity as you hold arrow keys
- Uses image sprites for objects in the game
- Objects rotate in addition to moving
  - `on_frame()` updates position, velocity, rotation, and angular velocity
  - `draw()` applies Transforms to objects
    - Place at position, Rotate to rotation, Scale based on mass
- Tracks key down/up to start and stop actions

Break + Sharing

- Come up with two possible project ideas that you think would work for CS211
  - You don’t have to actually do these!

- Share your ideas with someone nearby

Outline

- Final Project Overview
- Demo Games
- **Additional GE211 Functionality**
- Example: “snake game”

GE211 you’ve already used

- Abstract game class
  - draw(), on_frame()
- Events
  - Mouse and keys
    - Includes keyboard keys such as shift, ctrl, alt, and arrow keys
- Geometry
  - Posn, Rect, Dim
- Basic sprites
  - Rectangles and Circles of multiple colors

Additional GE211 Features

- **Resources files**
- Audio
- Advanced Sprites
- Sprite Manipulations
- Timer

Resources files

- Add a Resources/ directory to the project root
  - next to src/ and test/

- Put files into it that you want your game to access while running
  - Configurations
  - Level layouts
  - Images
  - Audio files

Accessing Resource files

- `ge211::open_resource_file(std::string const& filename)`
  - https://tov.github.io/ge211/namespacege211.html#a2dadd7cd96f1642d432e9d63de63f00c
  - Finds the filename specified and opens it for you
    - Don't specify `Resources/`, just the filename
  - Returns an `std::ifstream`

- Access the data within the `std::ifstream` with `>>`
  - Just like stdin

- Submitting Resources files:
  - Autograder puts everything that's not `*.cxx` or `*.hxx` into `Resources/`
  - Note: `test*.cxx` and `*test.cxx` go into `test/`

Additional GE211 Features

- Resources files
- **Audio**
- Advanced Sprites
- Sprite Manipulations
- Timer

Audio in GE211

- One Mixer controls all sounds for the game
  - https://tov.github.io/ge211/classge211_1_1audio_1_1_mixer.html

  - Can continuously play one Music_track (background music)
    - https://tov.github.io/ge211/classge211_1_1audio_1_1_music__track.html
    - play, pause, resume, rewind, set_volume

  - Can play short Sound_effects
    - https://tov.github.io/ge211/classge211_1_1audio_1_1_sound__effect.html
    - play, pause_all, resume_all
    - Can support several sound effects at once
      - Hardware dependent

Using audio

- How to get access to the mixer
    - Call `mixer()` inside the Controller
    - (Actually inside whatever inherits from Abstract_game)

- How to get a Music_track or Sound_effect
    - Call constructor with a filename string
    - Name of a file in Resources/
        - WAV, MP3, FLAC, MID, ABC, OGG, etc.

- Various sound effects and music can be found online

Additional GE211 Features

- Resources files
- Audio
- **Advanced Sprites**
  - **Text Sprites**
  - **Image Sprites**
- Sprite Manipulations
- Timer

Text Sprites

- Creates a sprite out of a string of text
  - Text, Color, and Font are configurable through a Builder
  - Placed on screen in `draw()` just like any other sprite
  - A little bit of work to manipulate though

- Text sprite can be reconfigured as needed
  - https://tov.github.io/ge211/classge211_1_1sprites_1_1_text__sprite.html

  - First use a Builder to create the text
    - https://tov.github.io/ge211/classge211_1_1sprites_1_1_text__sprite_1_1_builder.html

  - Then call `reconfigure()` with the Builder as the argument

Text sprite example

- Keep sprites and fonts as private members of View

```
unsigned int score;
ge211::Posn<int> score_position;
ge211::Font sans18{“sans.ttf”, 18};
ge211::Text_sprite score_sprite;
```

- In `draw()`, reconfigure the string as needed

```
ge211::Text_sprite::Builder current_score(sans18);
current_score << score;
score_sprite.reconfigure(current_score);
set.add_sprite(score_sprite, score_position);
```

Image Sprite

- `Image_sprite(std::string const& filename)`
  - Creates a sprite out of a given image
  - Uses the image's dimensions in pixels
    Transparency in images works!

- Filename comes from Resources/

Additional GE211 Features

- Resources files
- Audio
- Advanced Sprites
- **Sprite Manipulations**
- Timer

Applying Transforms to sprites

- What if your image sprite is larger than you want?
- Or if you want to rotate a sprite

- Transforms!
    - https://tov.github.io/ge211/classge211_1_1geometry_1_1_transform.html
    - Enable rotation, scaling, and flipping sprites

    - Passed in as an alternative final argument to draw()
    - https://tov.github.io/ge211/classge211_1_1_sprite___set.html#ad20a59df594c869b26e222da98c6161d

Additional GE211 Features

- Resources files
- Audio
- Advanced Sprites
- Sprite Manipulations
- **Timer**

Timers allow durations to be tracked

- Create a Timer() and start it
  - Later check it and you can see how long it was running
  - Allows you to determine how long some player action took
  - Probably NOT the right choice for most games (see next slide)

- Timer class
  - https://tov.github.io/ge211/classge211_1_1time_1_1_timer.html

  - Returns a Duration
    - https://tov.github.io/ge211/classge211_1_1time_1_1_duration.html
    - Which you can request time from in seconds or milliseconds

Easier way to track timing

- There’s an easier way to track time and perform actions after a certain amount of time has passed
- How would we use `on_frame(double dt)` to do so?
  - `dt` is in units of seconds
    - Usually 1/60th of a second
  - Keep a local variable that you add `dt` to each time `on_frame()` is called
    - Reset the variable to zero whenever you need to start counting
    - If variable is greater than some amount, trigger action

GE211 Examples

- Similar idea to the demo projects, but much simpler and cleaner

- Small snippets that only focus on a few ideas
  - Provides good reference code for how to use stuff

- Again, likely not useful as “starter code”, but you can use whatever code you want from these

GE211 example code

- https://nu-cs211.github.io/cs211-files/hw/ge211_examples.zip

- Includes three separate projects
  - sound
  - random_text
  - animation

Example: sound

- Plays a sound when the up arrow key is pressed
  - Also plays background music continuously

- Concepts
  - Resources/ audio files
  - Background music
  - Sound effects

Example: random_text

- Displays random words on screen in random colors at a random location

- Concepts
  - Resources/ text files
  - Text sprites
  - Transforms
  - Randomness

Example: animation

- Animates a character moving to wherever the mouse clicks
  - Keeps track of multiple mouse click locations
  - Spacebar pauses the game

- Concepts
  - Game states (init, running, paused)
  - Resources/ image files
  - Animation
  - Motion planning

Break + Request

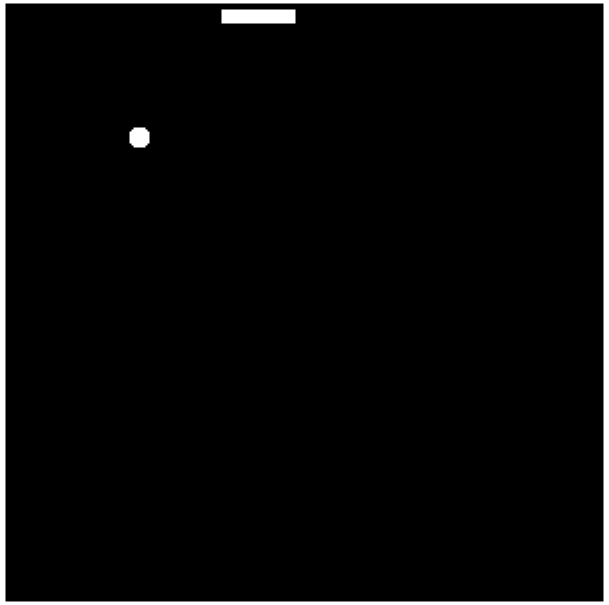
- Think of other things you might want to do with your project
  - But aren’t sure how to accomplish with GE211
- Share ideas with neighbors and talk about it
- If you come up with anything useful, share on Piazza!
  - I’m happy to give guidance and I could make more examples

Outline

- Final Project Overview
- Demo Games
- Additional GE211 Functionality
- **Example: “snake game”**

Multi-lecture project example

- Starting from https://nu-cs211.github.io/cs211-files/hw/final_project.zip
- We'll add features as we go
  - Probably not going to finish today
  - Plan to hop back into it in future lectures though
- Idea: Snake Game
  - Too simple for a final project
  - Simple enough to do in class?



Plan for game

- `List<Posn<int>>` for each "segment" of the snake
    - Consider the playing field as a 2D grid of locations
    - Posn<int> is one location on the grid
- Snake should "move" in current direction
    - Segment at end disappears
    - Segment at front gets added
    - Check for collisions
    - Occurs every N seconds?
- Draw each segment in the list to see the snake
- Key presses change direction of snake

Simplest initial design

- One segment only in the list

- Implement
  - Constructors
  - Model::on_frame() (most basic version)
  - View::draw()
  - Controller::on_key()

Start adding features

- Check for collisions
  - With body of snake
  - With edge of screen

- Resize draw based on screen dimensions and grid dimensions

- Goal object that increases snake length

- Obstacles to avoid

Outline

- Final Project Overview
- Demo Games
- Additional GE211 Functionality
- Example: “snake game”