CSD1130
Game Implementation Techniques

Lecture 3

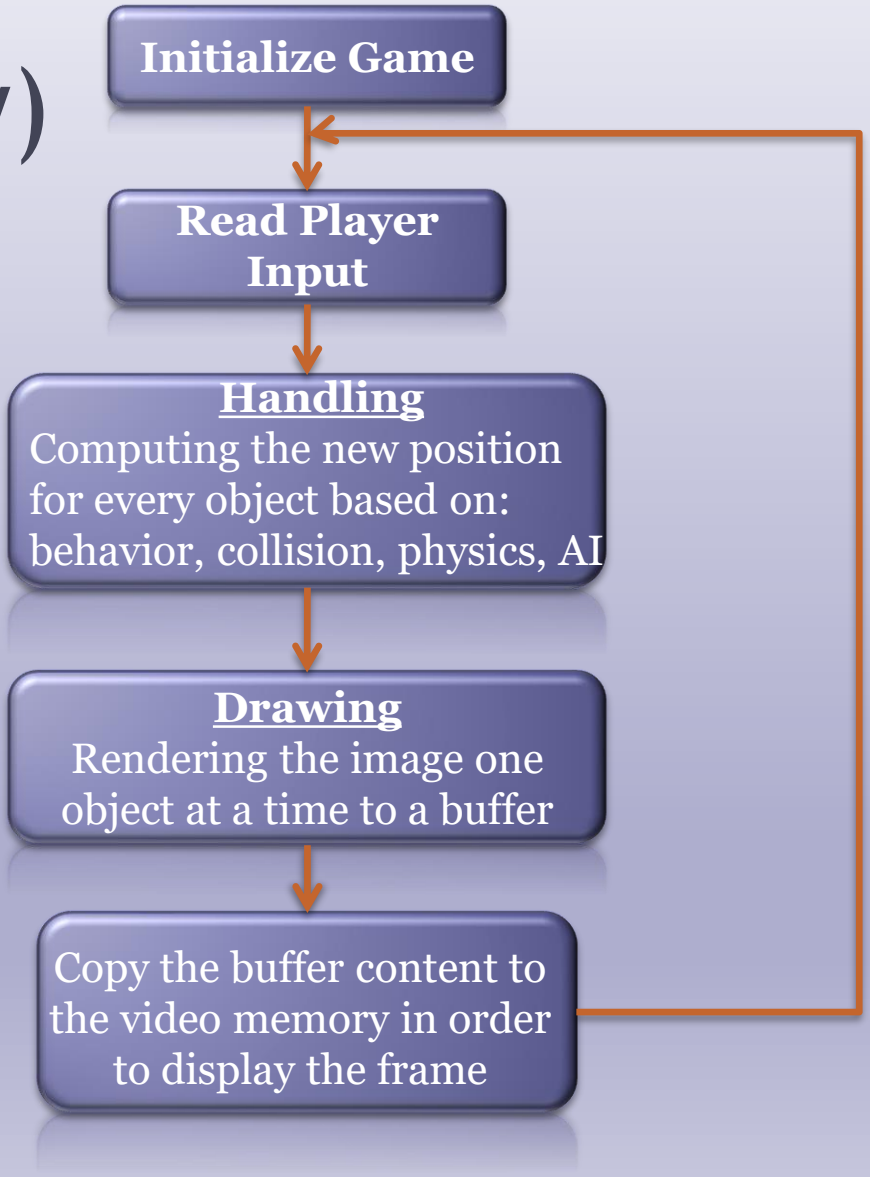
Questions?

- Game Engine Components
  - System Components
  - Game Logic Components

Overview

- Game State Manager
- Function Pointers

Game Loop (Review)



Game State Manager (GSM) (1/2)

- A game is always in a state. A game could be in “Main Menu”, in “Level 1”, in “Loading screen”…
- The GSM is responsible for game state switching, the game loop and the frame rate controller.
- Each state is associated with a set of functions that manages that state’s cycle.

Game State Manager (GSM) (2/2)

- The cycle functions are:
  - Load
  - Initialize
  - Update
  - Draw
  - Free
  - Unload

Cycle Functions: Load

- Loads the state's necessary data and initializes it.
  - Data here, represents the resources (assets) data
    - Example: Textures
- It is called once at the start of the state.
- It should NOT be called upon restarting a state.

Cycle Functions: Initialize

- Used to prepare the state's data in order to be used for the first time.
  - Data here, represents game objects instances (entities)
- It loads no assets whatsoever.
  - As an exception, the game objects instances can be loaded from files (serialization process)
- If a state is restarted, this cycle function is used.

Cycle Functions: Update & Draw

- Update:
  - Updates the state's data, based on several factors like user input, time or gameplay logic…
- Draw:
  - Sends data to the graphics engine component

Cycle Functions: Free

- Used to clean up the state.
  - Cleans the game objects instances
  - May clean (or save) scoring system
- Make the state ready to be unloaded or initialized again.
- No data is dumped in this cycle function

Cycle Functions: Unload

- Is called when the state should be terminated.
- It dumps back all the data that was loaded in the state's load cycle function.
  - Cleans the assets (free/release the memory)

Overview

- Game State Manager
- Function Pointers

Pointers to Functions

Check Uploaded Snippet Files (Pointers to Functions - MSVS.zip):

Snippet covers:

- Pointer to a function
- Array of function pointers
- Using “typedef” with function pointers
- Passing function pointers as arguments
- Returning function pointers