GPAT – CHAPTER 1 GAME PROGRAMMING OVERVIEW

BRIEF HISTORICAL REMARKS

CONSOLE FOCUSED

ATARI ERA (1977-1985)

- Very little RAM, slow processor speed
- All games created in assembly
- Solo programmers

NES AND SNES ERA (1985-1995)

- More powerful hardware, but not enough for C. However, developer kits were released to help debugging
- Small programmer teams (3-9)

PLAYSTATION/PLAYSTATION 2 ERA (1995-2005)

- More powerful hardware, single thread, single core
- Games written in C (assembly for performance critical sections)
- Early years 8-10 programmers, late years up to 15 programmers

XBOX 360, PS3, WII ERA (2005-2013)

- High definition support
- Advanced hardware support (multi-threading, multi-core)
- C++ and more with developer kits, e.g., Unity
- Programmer teams scaled immensely
  - Over 75 for Assassin's Creed Revelations, for example

XBOX ONE, PS4, WII SWITCH, AND BEYOND

- More cores, more memory, 4K resolution, and on
- Larger and larger teams
- More independent titles
  - Seeing resurgence of solo/small programmer teams

THE GAME LOOP

TRADITIONAL GAME LOOP

```
1. while gameIsRunning() do
2.    // Process inputs
3.    // Update game world
4.    // Generate outputs
```

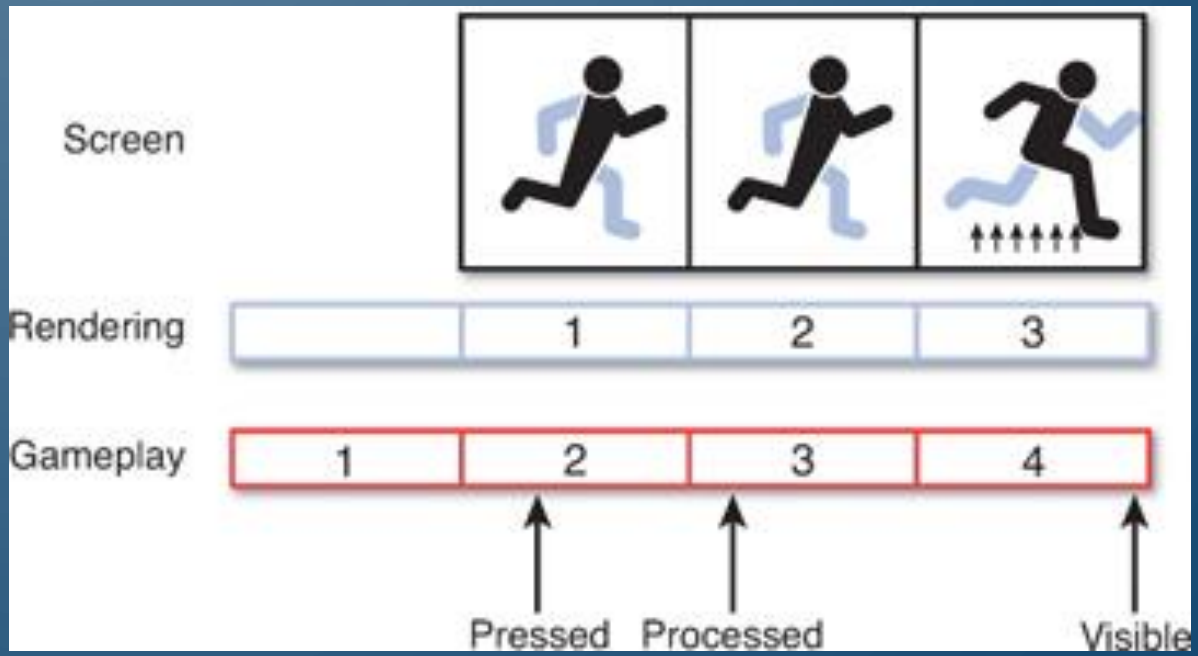
- Processing inputs requires detecting inputs from keyboard, mouse, controller, etc. Also includes communication over a network
- Generating outputs includes rendering graphics, audio, force feedback, etc.

EXAMPLE GAME LOOP FOR PAC-MAN

```
1. while player.lives > 0 do
2.   // Process inputs
3.   JoyStickData j = getJSD()
4.   // Update game world
5.   player.update(j)
6.   player.killOrDeath(ghosts)
7.   ghosts.updateAI(player)
8.   //   pellets, etc
9.   // Generate outputs
10.   drawWorld()
11.   updateAudio()
```

MULTI-THREADED GAME LOOPS

- More difficult to form to multi-core systems
- One such solution, separate rendering and delay by one frame
  - Creates **input lag**
- Other issues?



TIME

REAL TIME VS GAME TIME

- **Real time** is the amount of time passed in the physical world
- **Game time** is the amount of time elapsed in the imaginary world

- Considerations:
  - Pausing the game?
  - "bullet-time" physics?
  - Reverse time?
    - Example: Prince of Persia: The Sands of Time

LOGIC AS A FUNCTION OF DELTA TIME

- Early programming had a specific processor speed in mind, but once the processor speed was different the game would break
- **Delta time** is the amount of game time elapsed since the last frame
  - Think and program in a frame-centric and game-centric notion

GAME LOOP WITH DELTA TIME

```
1. while gameIsRunning() do
2.   realdt = lastt
3.   gamedt = realdt * gametf
4.   // Process inputs
5.   // Update game world with gamedt
6.   // Generate outputs
```

- Problems?
  - Different behavior with different frame rates
  - Online play?
- Solution – **frame limiting**, i.e., limit the frame rate

GAME LOOP WITH FRAME LIMITING

```
1.  targetft = 0.17f
2.  while gameIsRunning() do
3.    realdt = lastt
4.    gamedt = realdt * gametf
5.    // Process inputs
6.    // Update game world with gamedt
7.    // Generate outputs
8.    // Frame limiting
9.    while framet < targetft do
10.     doSomethingSmall()
```

- Problems?
  - **Dropping a frame**

GAME OBJECTS

TYPES OF GAME OBJECTS

- A **game object** is anything in the game world that needs to be updated, drawn, or both in every frame
  - Updateable and drawable
    - Example – Mario (or any character)
  - Drawable only
    - Example – Brick (or any **static object**)
  - Updatable only
    - Example – Camera, hit box, location that starts an event (**trigger**)

- How would you implement?
  - Interface for each + inheritance
- Class GameObject
- Interface Drawable
- Interface Updatable
- Classes for DrawableGameObject, UpdatableGameObject, DrawableUpdatableGameObject
- Class for GameWorld that contains lists of DrawableObjects and UpdatableObjects

GAME OBJECTS IN THE LOOP

```
1.  targetft = 0.17f
2.  while gameIsRunning() do
3.    realdt = lastt
4.    gamedt = realdt * gametf
5.    // Process inputs
6.    // Update game world
7.    for Updatable o in GameWorld.updateableObjects do
8.      o.update(gamedt)
9.    // Gemerate outputs
10.   for Drawable o in GameWorld.drawableObjects do
11.     o.draw()
12.   // Frame limiting
13.   while framet < targetft do
14.     doSomethingSmall()
```

SUMMARY

- Explored general frameworks surrounding game programming
- Need to remember to be frame-centric when developing a game