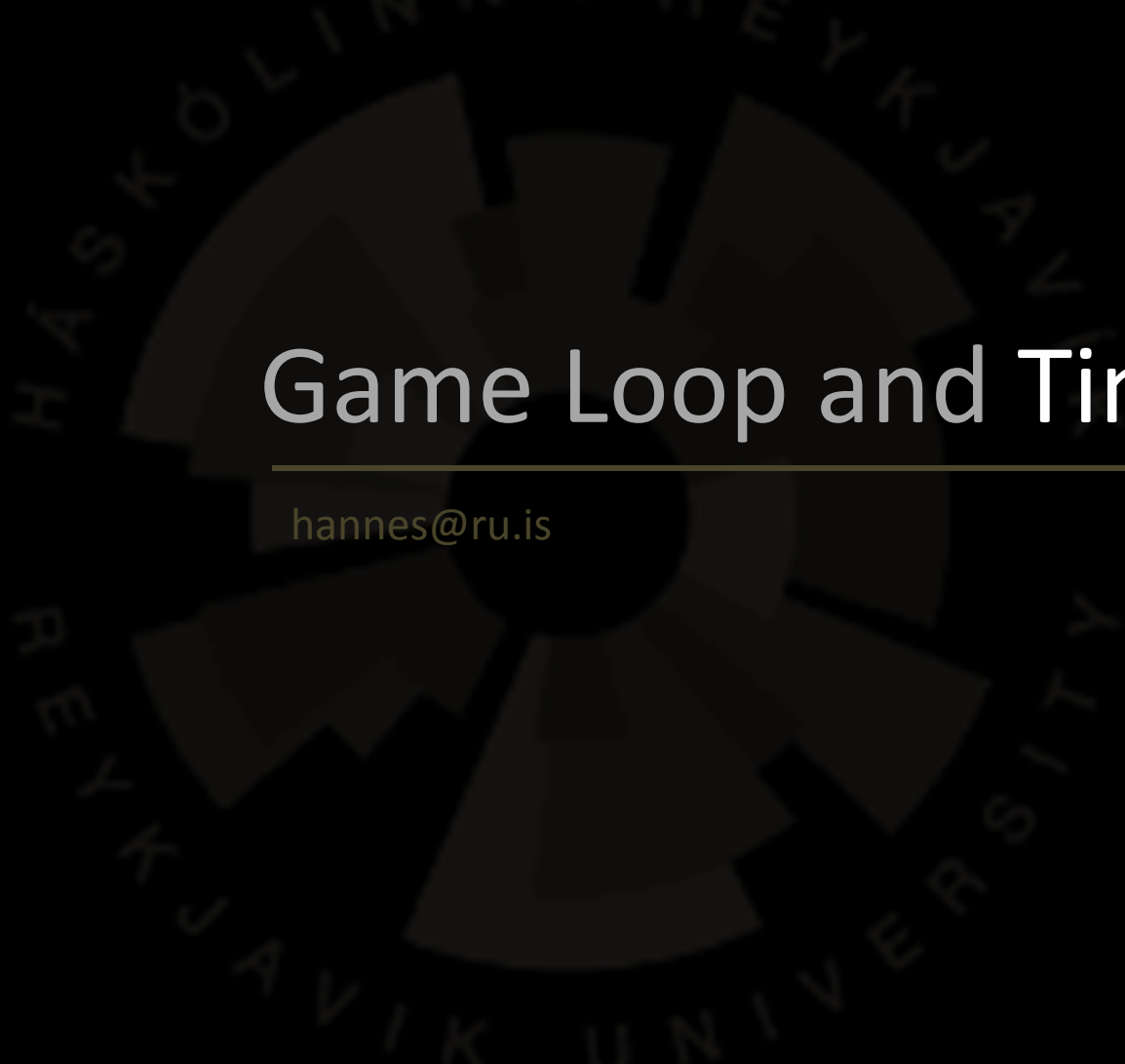

Game Loop and Time

hannes@ru.is

Importance of Time

- Games are...

  ...real-time, dynamic, interactive computer simulations

- It‘s all about time

Render Loop

- Not all applications need to redraw the whole screen image continuously
  - Windows uses method of „rectangle invalidation“
  - Early games used 2D sprites
- 3D games need to display a rapid succession of still images to produce a continuous experience
  - Accomplished with a „render loop“

Typical Render Loop

1. Update Camera
2. Update Scene Elements
3. Render Scene
4. Swap Buffers

Game Loop

- The game engine needs to service a range of subsystems
- Subsystems may require servicing at different rates

Architecture Styles

- Windows message pumps
  - Check and deal with win messages, then do game
- Callback-driven frameworks
  - Implement callback function called by framework
- Event-based updating
  - React to events, schedule recurring events

Abstract Timelines

- Real-time
  - CPU high-resolution time register
- Game time
  - May coincide w. real-time, can be paused/stepped

- Local timeline (e.g. video/animation playback)
- Global timeline (e.g. the real-time clock)

Measuring Time

- Frame Rate
  - How rapidly the still frame sequence is presented
- Frames Per Second (FPS)
  - Films: 24 FPS,
  - Games: 30/60 FPS (US/Japan), 50 FPS (EU)
- Delta Time
  - Time that elapses between showing two frames
  - At 30 FPS, delta time is 1/30 seconds or 33 ms

Speed

- We wish to render a car driving at 60 km/h

  .... or 50/3 m/s

- Change in position between drawings should be calculated, let‘s assume 30 FPS

  $\Delta x = \text{speed} * \Delta t = 50/3 \text{ m/s} * 1/30 \text{ s} = 5/9 \text{ m}$

- So, every time we draw, we put the car 5/9 m further along its trajectory

  works as long as we‘re getting 30 FPS...

Measuring Δt

- Update based on elapsed time
  - Δ t calculated when you finish a frame, made available to the next frame
  - But this time from frame k is used when calculating duration for the simulation at frame k+1
- Frame rate spikes can cause trouble
  - Use running average for Δ t instead

Frame Rate Governing

- Set target frame rate, e.g. 60 FPS
  - If duration of frame is less than 16.6 ms, then sleep for the remainder
  - If duration is greater, then so be it
- Useful because
  - Physics run more reliably at constant rate
  - Avoids tearing when buffers get swapped mid-scan
    - Frame rate can be governed by v-sync of display
  - Consistent playback of recorded play

Measuring Time

- time()
  - Seconds since midnight, January 1, 1970
  - Resolution 1 second (not enough!)
- High-resolution timer
  - CPU cycles since start-up or reset
  - Resolution 0.33 ns (for a 3 GHz processor)

High-Resolution Timer

- Hardware register
  - Queried with different instructions based on hardware platform
  - Win32 wraps these in QueryPerformanceCounter() and QueryPerformanceFrequency()
- Size of register limits magnitute
  - 64 bit register wraps after 195 years at 3 GHz
  - 32 bit register wraps after 1.4 seconds at 3 GHz

Time Units and Clock Variables

- When we wish to work with time, we ask:
  1. What time unit?
  2. What data type?
- The answers will depend on:
  1. How much precision?
  2. What magnitude range?

Time Variables

- 64 bit integer
  - High precision, high magnitude, but expensive
- 32 bit integer
  - High precision for short durations, e.g. Deltas
- 32 bit float
  - Store values in seconds
  - Careful about magnitude, it can eat up the decimals
  - Good for short time deltas (may need resetting)

Other Time Units

- Game developers can pick arbitrary time units if they wish
- Sometimes use 1/300 second units
  - Even multiple of NTSC 60 Hz and PAL 50 Hz
  - Fairly good sized unit for game related stuff, e.g. AI and object behavior

Break Points

- If you pause execution of program, e.g. for debugging, the Δ t when you start again will become super large, causing anomalies
- Fix this by checking in code whether Δ t is absurdly high, in which case a default value should be used instead