

# Lecture 4

## Frame Rate Controller

- |                        |   |
|------------------------|---|
| 1. Game Engine Design  | 2 |
| 1.1. System Components | 2 |

CS230  
Game  
Implementation  
Techniques

### Copyright Notice

Copyright © 2010 DigiPen (USA) Corp. and its owners. All rights reserved.

No parts of this publication may be copied or distributed, transmitted, transcribed, stored in a retrieval system, or translated into any human or computer language without the express written permission of DigiPen (USA) Corp., 9931 Willows Road NE, Redmond, WA 98052

### Trademarks

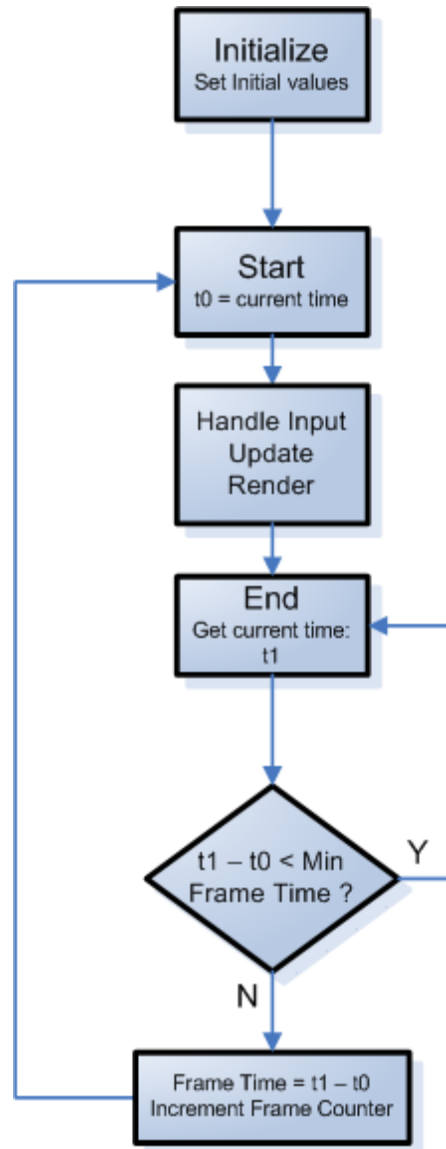
DigiPen® is a registered trademark of DigiPen (USA) Corp.

All other product names mentioned in this booklet are trademarks or registered trademarks of their respective companies and are hereby acknowledged.

# 1. Game Engine Design

## 1.1. System Components

- **Game state manager**
  - (Explained in previous session)
- **Frame rate controller**
  - In games, the amount of time needed to complete 1 frame (That is updating the objects, updating AI, checking for collision, reflection, physics, checking/updating input, updating general gameplay, rendering...) is extremely inconsistent. It depends on many factors like:
    - Detail level
    - Number of visible objects
    - Number of effects (lights, normal mapping, shadows...)
    - AI depth
    - Collision types (Precise collision checks take more time than basic collision checks)
    - Physics iterations
    - Etc...
  - The main job of a frame rate controller is to make sure the game is running at a consistent speed.
  - This is done by making sure each frame is taking a certain pre-set amount of time.
  - Simply, whenever a frame is completed and it's time to go back to the beginning of the game loop (which is the beginning of the next frame or loop), we check how much time the current frame took.
  - If it's less than the pre-set amount, then we keep the game “waiting” - which is basically creating an empty loop – until that amount of time is reached.
  - On the other hand, if the amount of time taken to complete 1 frame is already greater than the wanted amount, then the way to fix it is by doing some optimizations (Like avoiding rendering objects outside the viewport's boundaries), reduce the number of effects used..



- The frame rate controller can have other secondary functionalities like:
  - The number of frames since the game started
  - The number of frames since the current game state started
  - How much time the last frame used (This info could be used by time dependent objects, like animations and physics)
- Since the frame rate controller needs to know the time at the beginning and end of each loop, its functions should be called at these points.

Frame rate controller pseudo code:

Initialization

Frames counter	= 0
Max Frame Rate	= 60
Frame Rate	= Max Frame Rate
Frame Time	= $1 / \text{Frame Rate}$
Min Frame Time	= $1 / \text{Max Frame Rate}$

Frame Controller Start

Frame Start	= Current Time
-------------	----------------

Frame Controller End

While((Current Time – Frame Start) < (Min Frame Time))

Frame End	= Current Time
-----------	----------------

Frame Time	= Frame End – Frame Start
------------	---------------------------

Increment Frame Counter

- Note that the time received from the system (Current Time in the pseudo code) should have at least a 1 millisecond precision in order for the frame rate controller to work.
- This is the game flow chart after implementing a frame rate controller:

