CS529
Fundamentals of Game Development

Lecture 3a

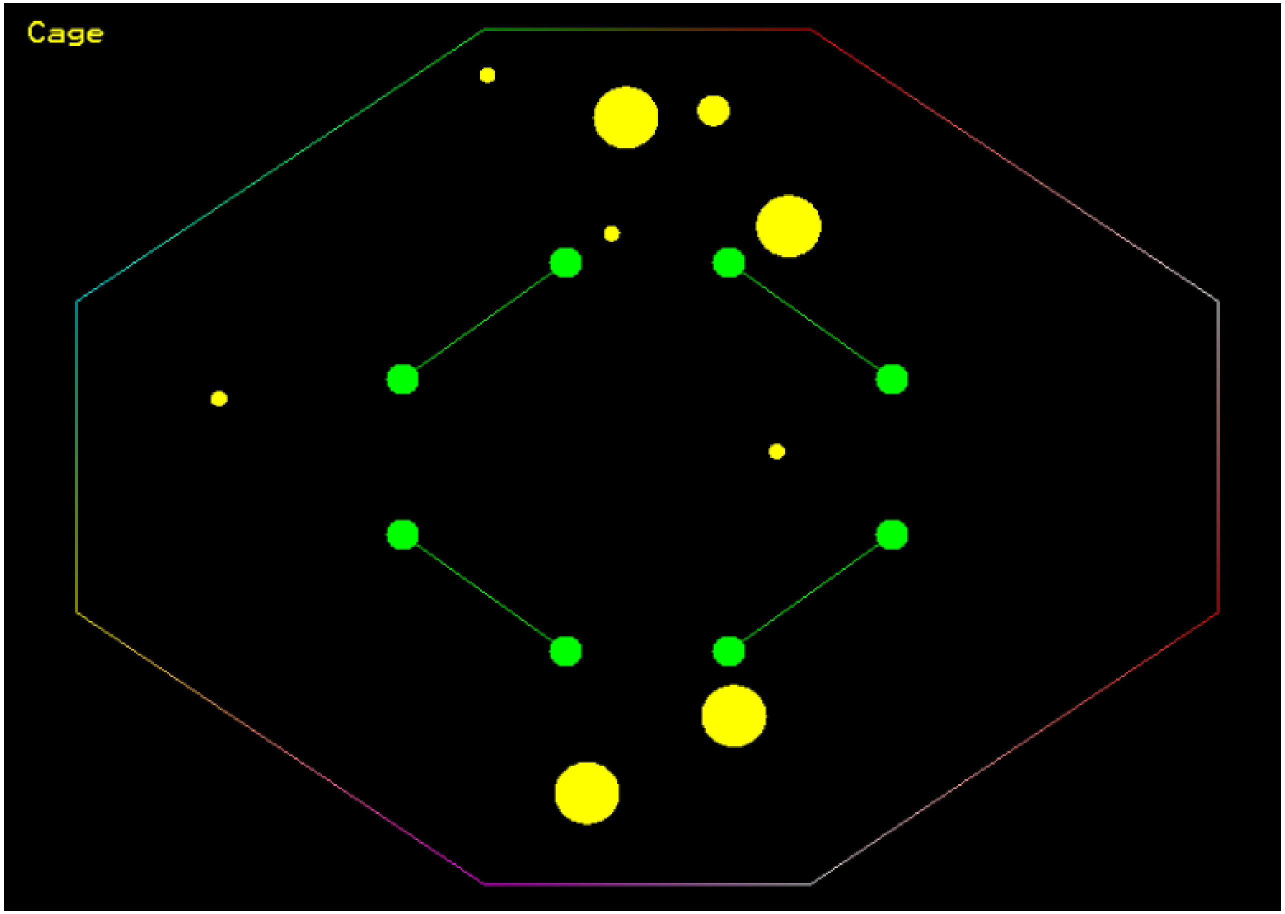
Antoine Abi Chakra
Karim Fikani

Questions?

- Windows Programming
  - Introduction
  - WinMain
  - Window Class
  - Create/Show the Window
  - Handling Events (Window Procedure Function)
  - Event (Message) Loop

Overview

- Object Kinematics
- Game Loop Review
- Object Animation


Cage

Object Kinematics (1/2)

- A CS529 object has a position and a velocity
  - ▫ Objects do not respond to forces
  - ▫ Objects move with constant velocity – that is, zero acceleration
  - ▫ Simplest to simulate

Object Kinematics (2/2)

- Obvious structure definition in C might look like (neglecting appearance and other properties):

```
struct Object
{
       … // Object methods and variables
       float  p[2];    // Position
       float  v[2];    // Velocity
};
```

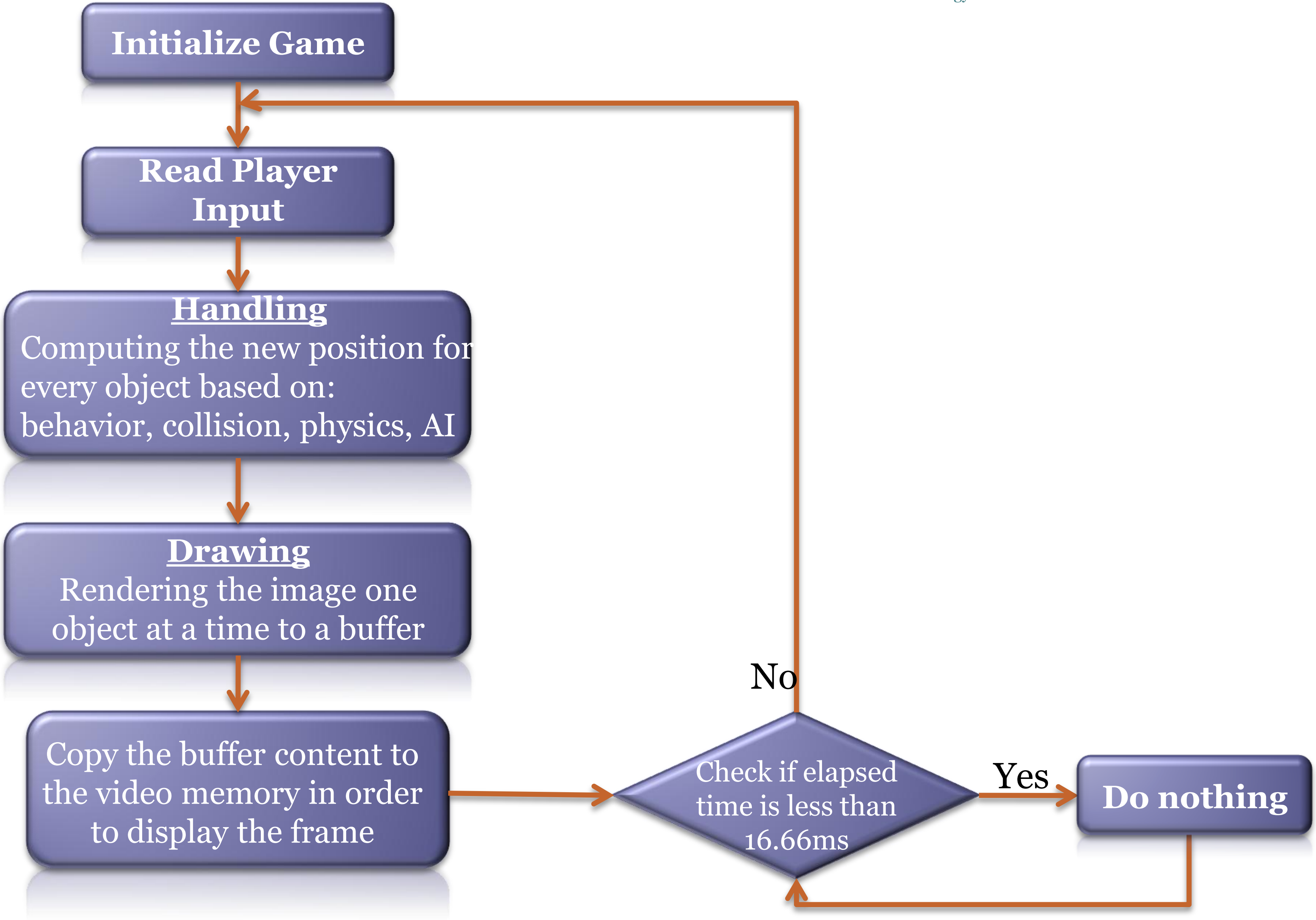
Object Animation

- Specify the initial position *p* and velocity *v* of each object
  - Velocity consists of a speed and direction vector (that is, a vector with unit magnitude)

- Every frame, update object's previous position:

$$\vec{\boldsymbol{p}} += \vec{\boldsymbol{v}}$$

- This movement type is called "Frame Based"



Initialize Game

Read Player Input

Handling
Computing the new position for every object based on: behavior, collision, physics, AI

Drawing
Rendering the image one object at a time to a buffer

Copy the buffer content to the video memory in order to display the frame

Check if elapsed time is less than 16.66ms

Do nothing

Better Game Loop (1/2) - Revisited

- Objects are no longer updated based on a pre-determined time between successive frames
- Instead, time interval to complete current frame is used in kinematics calculations to determine objects' displacements
  - Computing time interval to complete current frame is non-trivial problem
  - Instead, good compromise is to use time interval of previous frame

Better Game Loop (2/2) - Revisited

```
double t = 0.0f; // game time (in seconds)
double currTime = time( ); // measure time at start of frame
Initialize_Game_Objects( t, 0.0f );
Draw_Game_Objects( );

while (!quit)
{
   double newTime = time( ); // measure time at end of previous frame or time at
   start                                // of current frame
   double dt = newTime - currTime; // time interval for previous frame (in seconds)
   currTime = newTime; // time at start of current frame
   Update_Game_Objects( t, dt );
   Draw_Game_Objects( );

   // Lock the frame rate here

   t += dt; // update game time with time interval of previous frame
}
```

Object Animation (Revisited) (1/6)

- Specify the initial position *p* and velocity *v* of each object
  - Velocity consists of a speed and direction vector (that is, a vector with unit magnitude)

Object Animation (Revisited) (2/6)

- Each frame:
  - Compute time interval between previous and current frame: dt
  - Compute object's displacement within time interval *dt:* $\vec{v} * dt$
  - Finally, compute object's new position as

$$\vec{\boldsymbol{p}} += \vec{\mathbf{v}} * \boldsymbol{dt}$$

- This movement type is called "Time Based"

Object Animation (Revisited) (3/6)

$$\vec{p} += \vec{v}$$

- Example: Frame based
- Velocity is: $\vec{v} = (3, 0)$
  - ▫ At 60 FPS, the object will move 180 units per second
  - ▫ At 30 FPS, the object will move 90 units per second
  - ▫ At X FPS, the object will move 3*X units per second

Object Animation (Revisited) (4/6)

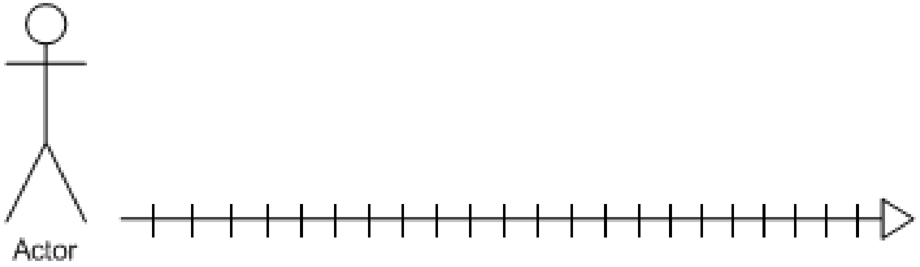
$$\vec{p} += \vec{v} * dt$$

- Example: Time based
- Velocity is: $\vec{v} = (180, 0)$

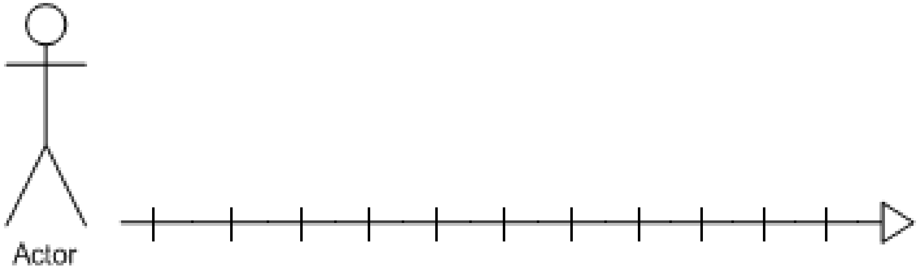
  - At 60 FPS, the object will move 180 units per second
  - At 30 FPS, the object will move 180 units per second
  - At X FPS, the object will move 180 units per second
    - Assuming X is not equal to 0

Object Animation (Revisited) (5/6)

- In time based games, the step size will adjust according to the frame time
- 60 FPS:



- 30 FPS:



Object Animation (Revisited) (6/6)

- Conclusion:

  In a time based application, given a time period, an animated object will always reach the same position, independently from the game's frame rate. What differs is the smoothness of the movement, where a slow FPS will make the character look as if it's disappearing and reappearing at its new location (Which is technically true!) instead of creating the illusion of motion.