Computer Graphics: Interaction (Camera)

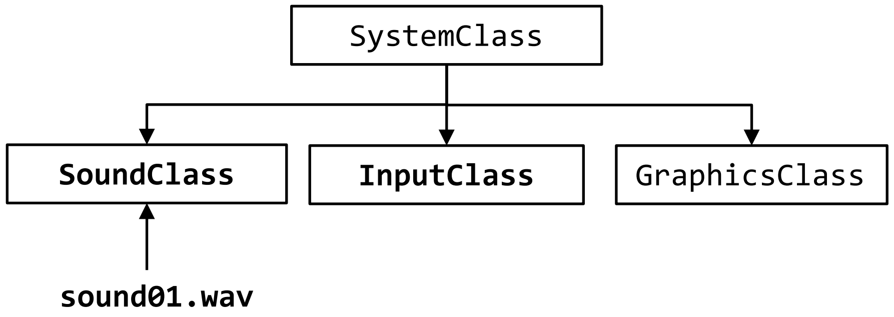
Dept. of Game Software
Yejin Kim

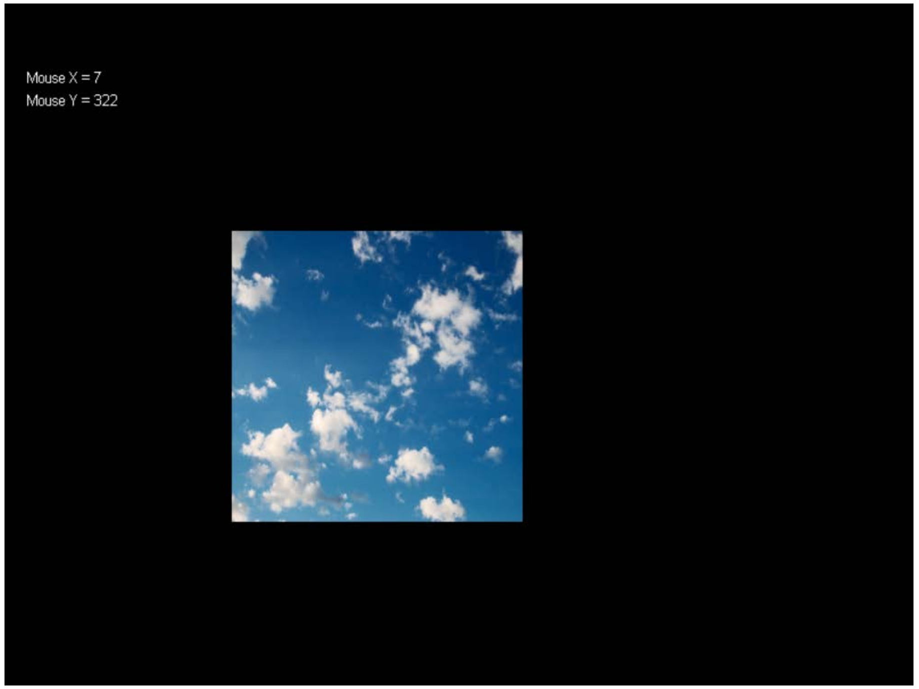
Tutorials

- DirectX Input and Sound
- First Person Camera
- Free Look Camera
- Rendering Information

6-1 Direct Input and Sound

- Adding Direct input and sound to the framework
  - InputClass: accepts input devices
  - SoundClass: loads and plays WAV audio files





```
Mouse X = 7
Mouse Y = 322
```

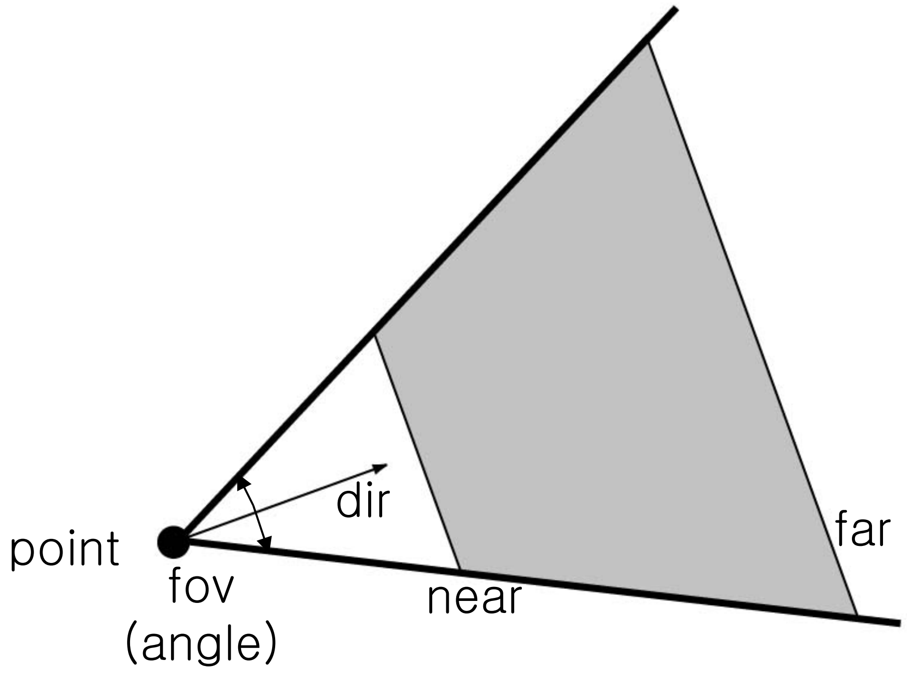
6-1 Direct Input and Sound

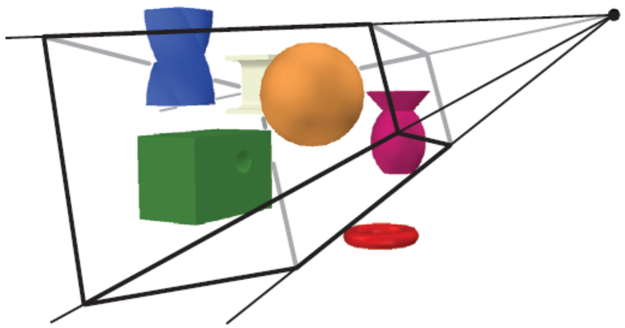
- Initializing Direct input devices: keyboard and mouse
  - DirectInput8Create()
  - IDirectInput8::CreateDevice()
  - IDrectInputDevice8::SetDataFormat()
  - IDrectInputDevice8::SetCooerativeLevel()
  - IDrectInputDevice8::Acquire()
  - IDrectInputDevice8::Unacquire()
  - IDrectInputDevice8::Release()

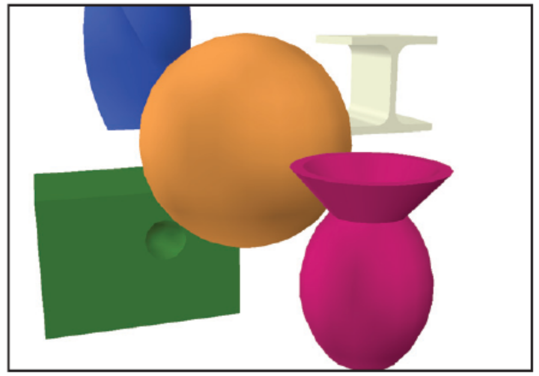
- Use WAV audio files
  - Should use WAV format: 44.1KHz, 16bit, 2 channels
  - Do **not** use a web-based converter for sound files: MP3 → WAV

6-2 First Person Camera

- Creating the first person camera (*BraynzarSoft)
  - Defined by position, direction vector, up vector, field of view, near and far plane
  - Create image of geometry inside gray region
  - Used by OpenGL, DirectX, ray tracing, etc







6-2 First Person Camera

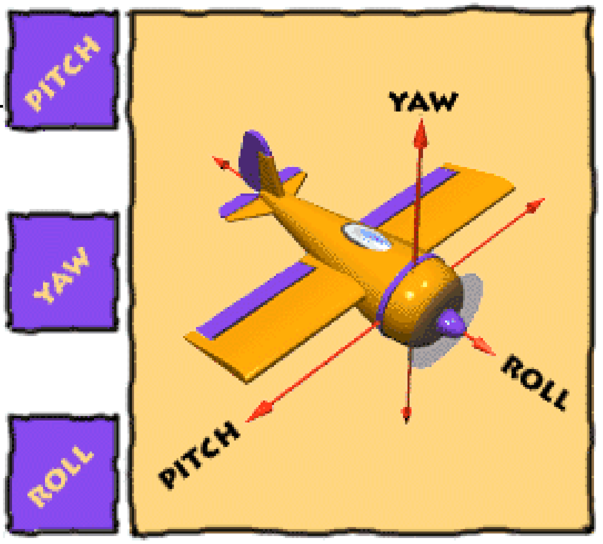
- Creating the first person camera (*BraynzarSoft)

```
XMVECTOR DefaultForward = XMVectorSet(0.0f,0.0f,1.0f, 0.0f);// the forward direction in the world
XMVECTOR DefaultRight = XMVectorSet(1.0f,0.0f,0.0f, 0.0f);  // the right direction in the world
XMVECTOR camForward = XMVectorSet(0.0f,0.0f,1.0f, 0.0f);    // the forward direction of the camera
XMVECTOR camRight = XMVectorSet(1.0f,0.0f,0.0f, 0.0f);      // the right direction of the camera

XMMATRIX camRotationMatrix;          // the rotation matrix of the camera
XMMATRIX groundWorld;                // the world matrix of the ground plane

float moveLeftRight = 0.0f;          // to move the camera strafe right/left
float moveBackForward = 0.0f;        // to move the camera forward/backward

float camYaw = 0.0f;                 // rotation around the y-axis
float camPitch = 0.0f;               // rotation around the x-axis
```



6-2 First Person Camera

- Creating the first person camera (*BraynzarSoft)

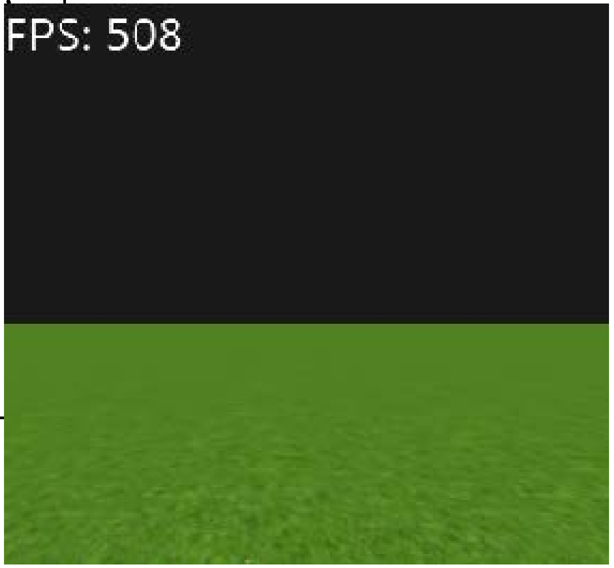
```
void UpdateCamera() {
      // Rotating the camera
      camRotationMatrix = XMMatrixRotationRollPitchYaw(camPitch, camYaw, 0);
      camTarget = XMVector3TransformCoord(DefaultForward, camRotationMatrix );
      camTarget = XMVector3Normalize(camTarget);

      // Restricting the camera rotation around the y-axis
      XMMATRIX RotateYTempMatrix;
      RotateYTempMatrix = XMMatrixRotationY(camYaw);

      // Updating the camera's right, up, and forward vectors
      camRight = XMVector3TransformCoord(DefaultRight, RotateYTempMatrix);
      camUp = XMVector3TransformCoord(camUp, RotateYTempMatrix);
      camForward = XMVector3TransformCoord(DefaultForward, RotateYTempMatrix);

      // Moving the camera
      camPosition += moveLeftRight*camRight;
      camPosition += moveBackForward*camForward;
      moveLeftRight = 0.0f;
      moveBackForward = 0.0f;

      // Updating the camera matrix
      camTarget = camPosition + camTarget;
      camView = XMMatrixLookAtLH( camPosition, camTarget, camUp );
}
```



6-3 Free Look Camera

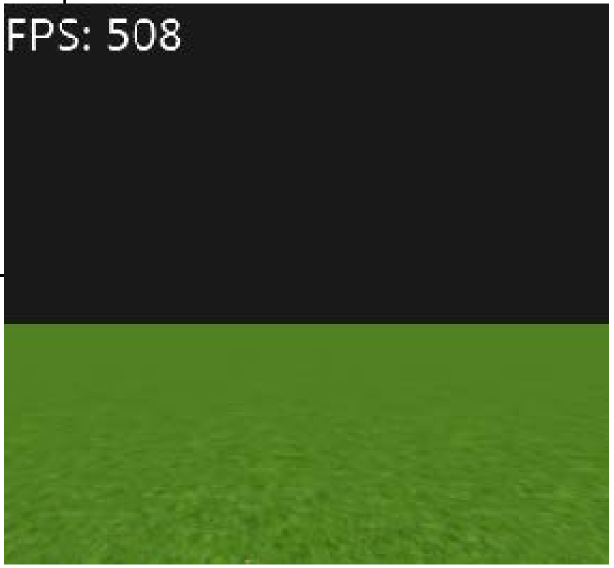
- Creating the free look camera (*BraynzarSoft)

```
void UpdateCamera() {
      // Rotating the camera
      camRotationMatrix = XMMatrixRotationRollPitchYaw(camPitch, camYaw, 0);
      camTarget = XMVector3TransformCoord(DefaultForward, camRotationMatrix );
      camTarget = XMVector3Normalize(camTarget);

      // Updating the camera's right, up, and forward vectors
      camRight = XMVector3TransformCoord(DefaultRight, camRotationMatrix);
      camForward = XMVector3TransformCoord(DefaultForward, camRotationMatrix);
      camUp = XMVector3Cross(camForward, camRight);

      // Moving the camera
      camPosition += moveLeftRight*camRight;
      camPosition += moveBackForward*camForward;
      moveLeftRight = 0.0f;
      moveBackForward = 0.0f;

      // Updating the camera matrix
      camTarget = camPosition + camTarget;
      camView = XMMatrixLookAtLH( camPosition, camTarget, camUp );
}
```



6-4 Rendering Information

- Timer: Real-time estimation
  - System time을 알려주는 함수 사용 (C언어)

    time();

    - 현대 game에선 정확성이 부족한 경우가 많음
    - e.g. The number of *seconds* since midnight, Jan 1, 1970
  - High-resolution timer (정밀 타이머) 사용 (Win32 APIs)

    QueryPerformanceCounter();   // read the counter (64bit integer)

    QueryPerformanceFrequency();   // number of cycles per second

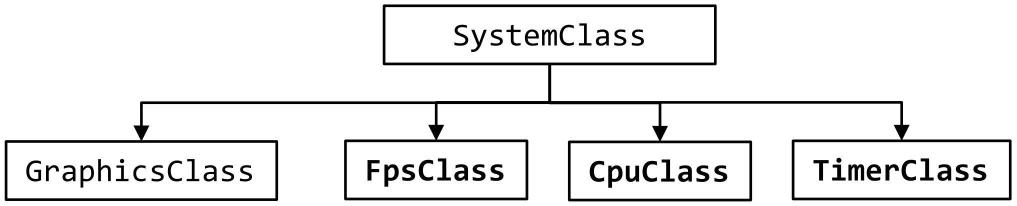
    - CPU에 power가 reset된 시점부터 초당 cycle의 수를 측정
    - e.g. 3GHz CPU의 경우 초당 30억분의 1로 시간을 나눌 수 있음

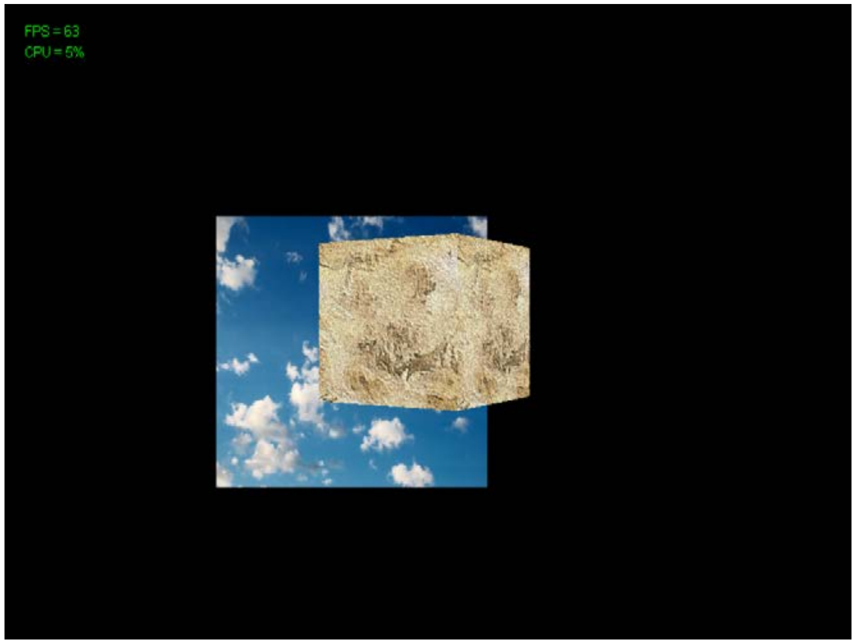
      정밀도: 1/(30억) = 3.33 x 10^{-10} sec = 0.333 ns

      64-bit int register를 timer로 사용시 reset되는데 195년 걸림

6-4 Rendering Information

- Draw FPS and CPU usage on screen
  - TimerClass: a high precision timer that measures the exact time between frames of execution
  - FpsClass: counts and updates frame numbers
  - CpuClass: estimates the total CPU usage




FPS = 63
CPU = 5%

References

- Wikipedia
  - www.wikipedia.org
- Introduction to DirectX 11
  - www.3dgep.com/introduction-to-directx-11
- Raster Tek
  - www.ratertek.com
- Braynzar Soft
  - www.braynzarsoft.net)
- CS 445: Introduction to Computer Graphics *[Aaron Bloomield]*
  - www.cs.virginia.edu/~asb/teaching/cs445-fall06

Q & A