DM 2231
GAMES DEVELOPMENT TECHNIQUES

2015/16 SEMESTER 1

Week 5 – Camera and GUI #2

MODULE SCHEDULE

Week	Dates	Topic	Remarks	Public Holidays
1	20-Apr-2015 to 24-Apr-2015	Module Introduction / 3D Game Programming	Issue Assignment 1	
2	27-Apr-2015 to 1-May-2015	Game Application		1 May. Labour Day
3	4-May-2015 to 8-May-2015	User Input		
4	11-May-2015 to 15-May-2015	Camera and GUI #1		
5	18-May-2015 to 22-May-2015	Camera and GUI #2		
6	25-May-2015 to 29-May-2015	Basic Game Physics		
7	1-Jun-2015 to 5-Jun-2015	Implementing Game Audio (E-learning)	Submit Assignment 1	1 Jun. Vesak Day
8	8-Jun-2015 to 12-Jun-2015	Mid-Sem Break		
9	15-Jun-2015 to 19-Jun-2015	Mid-Sem Break		
10	22-Jun-2015 to 26-Jun-2015	2D Game Programming #1	Issue Assignment 2	
11	29-Jun-2015 to 3-Jul-2015	2D Game Programming #2		
12	6-Jul-2015 to 10-Jul-2015	2D Game Programming #3		
13	13-Jul-2015 to 17-Jul-2015	Game Data		17 Jul. Hari Raya Puasa
14	20-Jul-2015 to 24-Jul-2015	Design Pattern #1		
15	27-Jul-2015 to 31-Jul-2015	Design Pattern #2		
16	3-Aug-2015 to 7-Aug-2015	Basic Artificial Intelligence (E-learning)		7 Aug. SG50 Public Holiday
17	10-Aug-2015 to 14-Aug-2015	Good Programming Practices	Submit Assignment 2	10 Aug. National Day

RECAP ON LAST WEEK'S LECTURE

- We have discussed about the main issues with Camera Control
  - The role of cameras in video games
  - Using a Camera Class to encapsulate camera movement methods and View Matrix
  - First-Person Shooter camera
  - Camera Inertia

TABLE OF CONTENT

- Camera and GUI #2
  - Third-Person Cameras
  - HUD
  - Minimaps

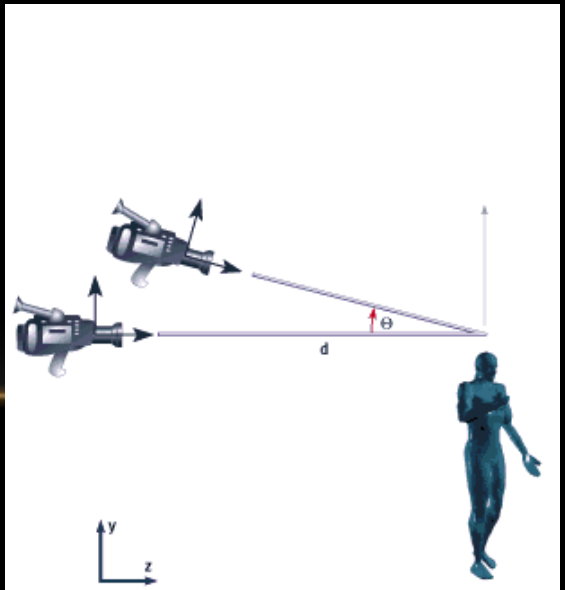
CAMERA: THIRD-PERSON CAMERA

- Third person games like Tomb Raider titles uses a floating camera that follows the player behind and above his head.
- Camera is placed behind the player
  - At an elevation angle over his position.
- The camera looks at the player,
  - It occupies the area below the center of the screen.

CAMERA:
THIRD-PERSON CAMERA

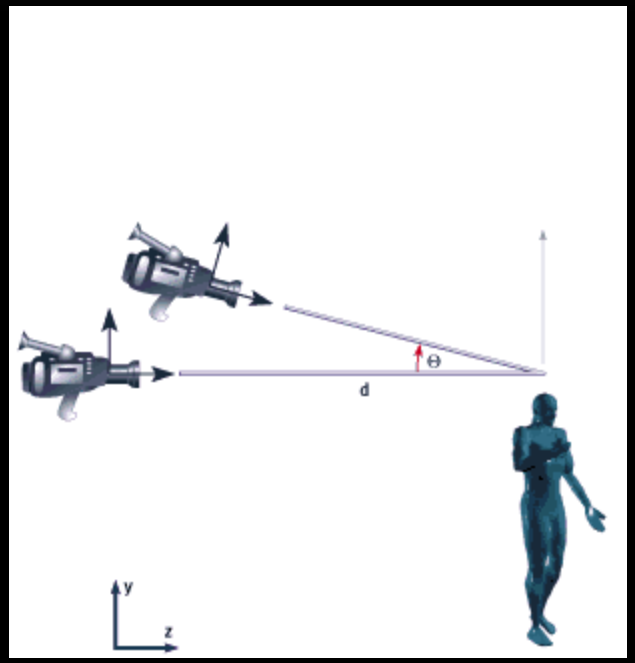
- The equations for this first camera's position and look-at point:

```
camPos.x= playerpos.x – cos(yaw)*cos(pitch)*distance;
camPos.y= playerpos.y + sin(pitch)*distance
camPos.z= playerpos.z – sin(yaw)*cos(pitch)*distance;
camlookat=playerpos;
```



CAMERA: THIRD-PERSON CAMERA

- Note that *pitch=0* means the camera is at the same height as playerpos
- Limit the camera to a pitch no greater than *PI/2 (90°)*
  - Otherwise, the camera will be upside down because we would have surpassed the vertical (90° from ground level).



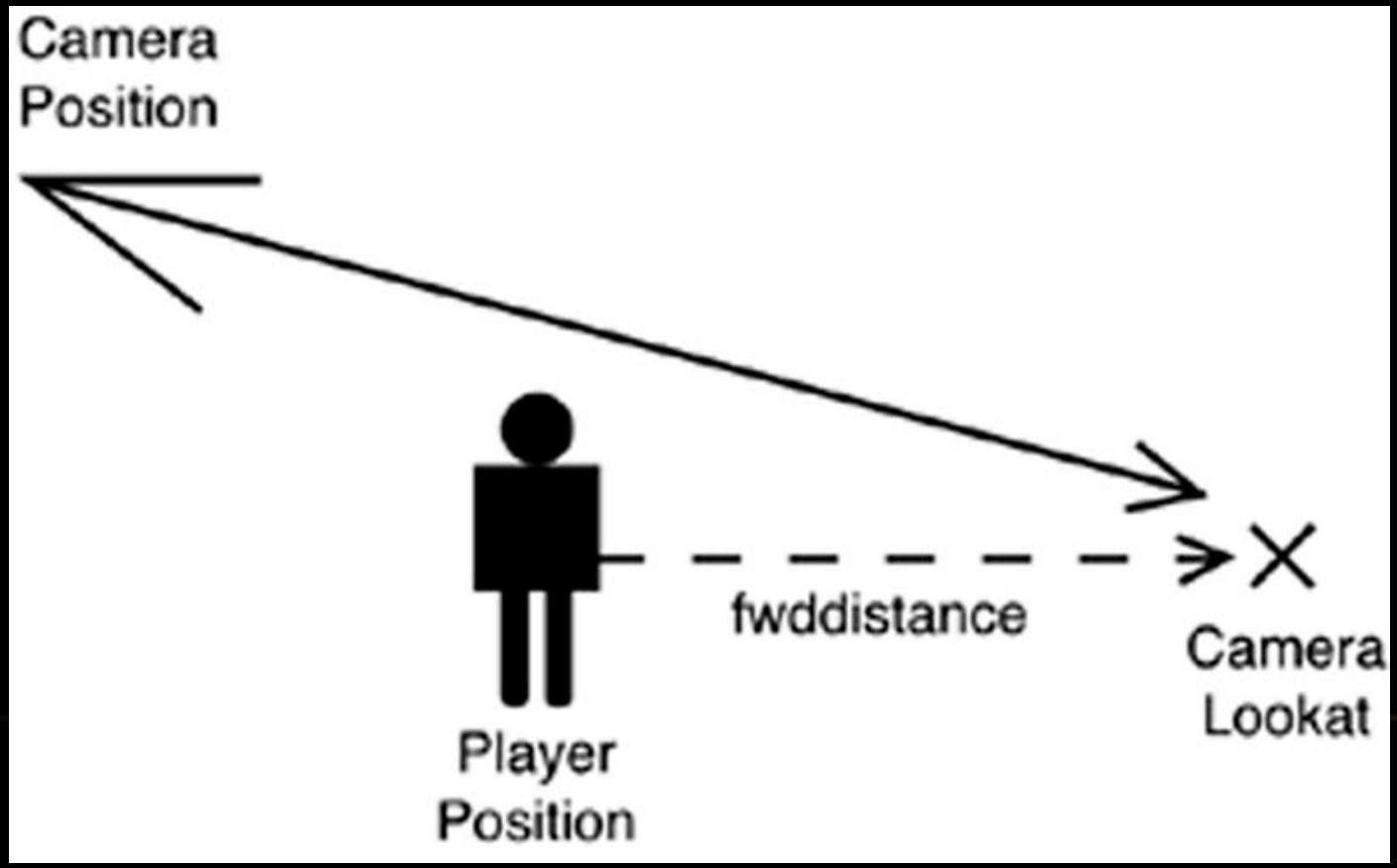
CAMERA:
THIRD-PERSON CAMERA

- Currently, the camera looks at the player, and he will occlude the enemy
  - Let’s improve the preceding code
    - so we do not aim our camera at the player directly, but in front of him.
  - This will make the player move vertically to the lower part of the screen,
    - Ensures we get a clear vision of whatever he's facing.

CAMERA: THIRD-PERSON CAMERA

- We change the look-at point:

```
camlookat=playerpos + playerdir * distance;
```



CAMERA:
THIRD-PERSON CAMERA

- Problem!
  - This camera model causes motion sickness.
  - Rotate player == Camera moving in wide arcs
    - Moving the camera too fast == motion sickness
- Solution:
  - Implement an inertia effect into the camera movement

CAMERA:
THIRD-PERSON CAMERA

- The idea is to…
  - limit the speed of the camera, and
  - use the position and look-at values computed previously only as indications of where the camera is moving to, not where the camera really is.
- When doing inertial cameras, we need to implement smooth interpolations between different orientations.

CAMERA:
THIRD-PERSON CAMERA

- When doing rotations using Euler, there's Gimbal Lock problem.
  - Lose one degree of freedom.
  - Solution: Use quaternions instead of Euler.
    - Quaternions provide an intuitive, simple mechanism to interpolate orientations.
    - Uses complex numbers
      - Less Computational Overhead
      - Better keyframe interpolation

CAMERA:
THIRD-PERSON CAMERA

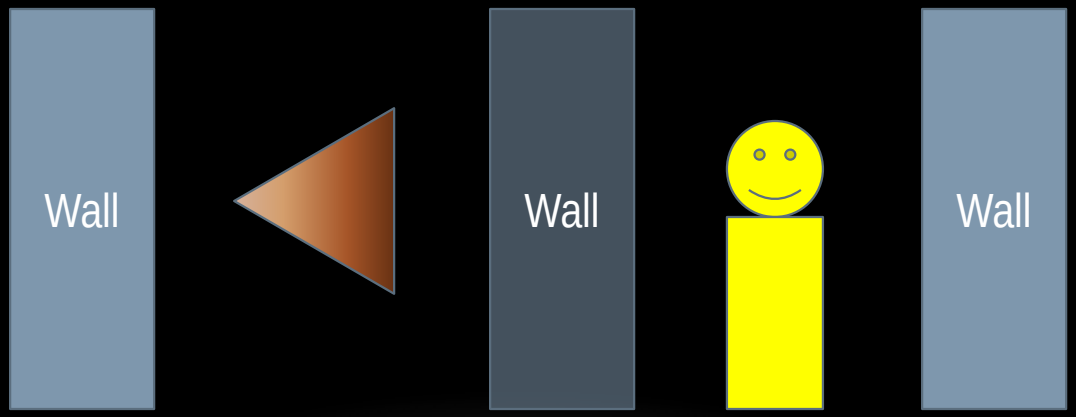
- Another problem:
  - What if the camera collides with surrounding objects?
    - Imagine that your character moves backward, so his back ends up leaning against a wall – the camera will actually cross the wall, and the sense of realism will be destroyed.

CAMERA:
THIRD-PERSON CAMERA

- Solutions:
  - First option: Let the camera cross the wall, but never allow this geometry to occlude what's going on.
    - Geometric objects between the camera and the player are alpha-blended, so room walls become partially transparent.
    - This is a relatively straightforward effect, but may not appear very convincing to the player.

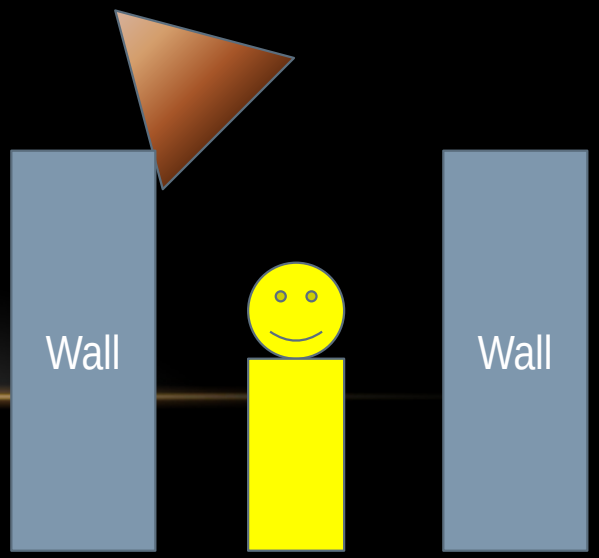
CAMERA:
THIRD-PERSON CAMERA

- Solutions:
  - First option: Let the camera cross the wall, but never allow this geometry to occlude what's going on.



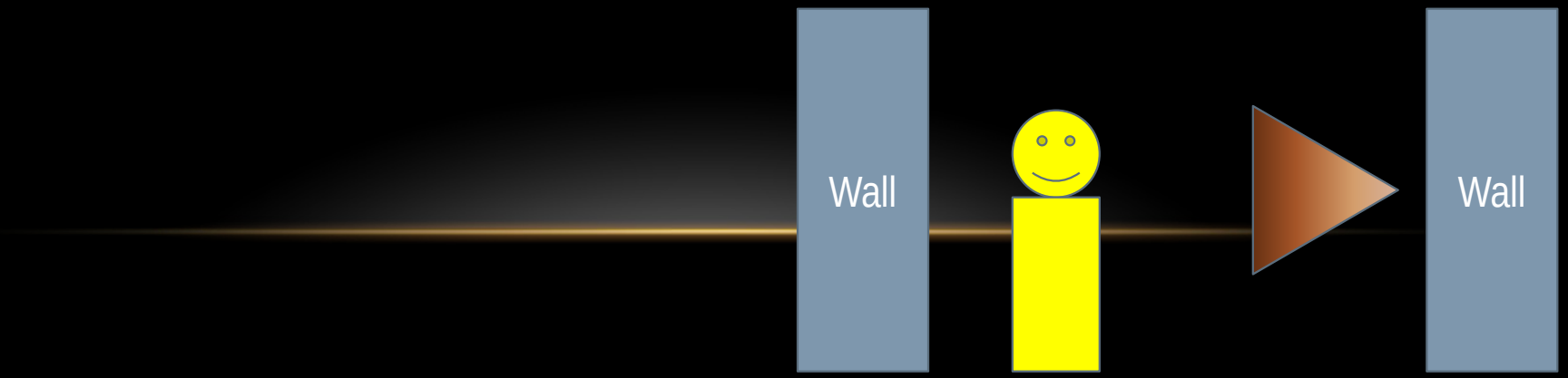
CAMERA:
THIRD-PERSON CAMERA

- Second option: Seek alternative camera positions if we detect we are about to cross level geometry.
  - Common approach
  - Finding the right spot is not an easy task.
    - Raise the camera vertically, but may lose some perspective on what's coming towards the player.
      - What if the room has a low roof?



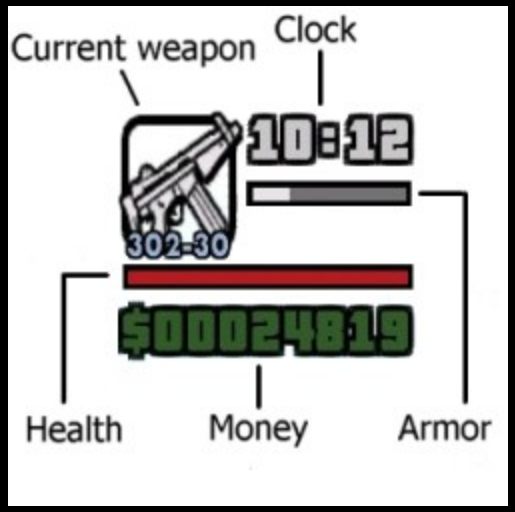
CAMERA:
THIRD-PERSON CAMERA

- Place the camera laterally
  - What if the player is at a corner or anywhere with geometry to the sides?
- Inverted shot: Instead of shooting from behind the player, we can place the camera in front of him
  - so we actually see the enemies approaching.



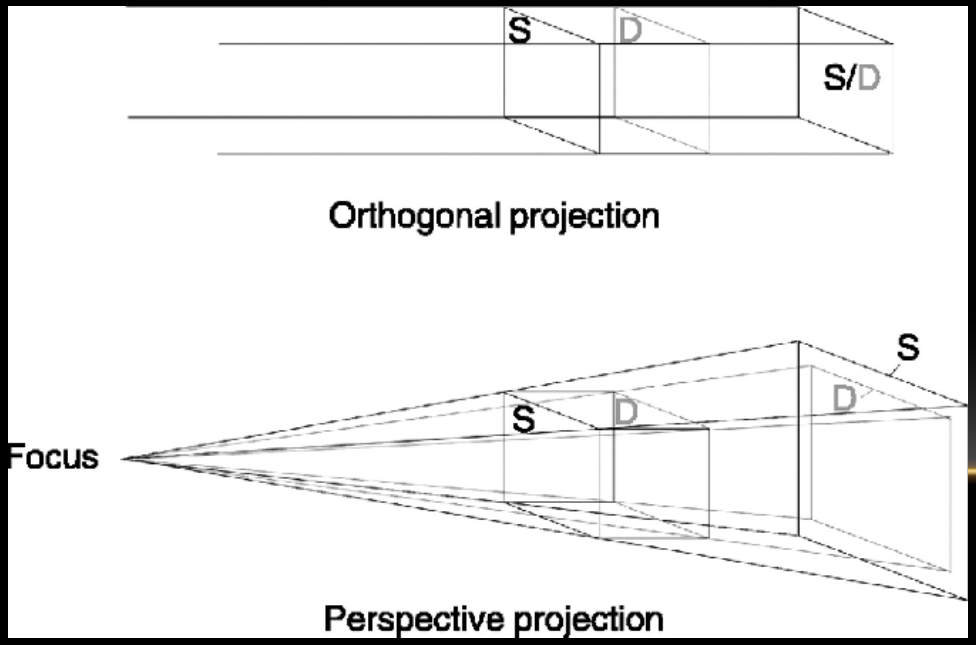
CAMERA:
HEADS UP DISPLAY

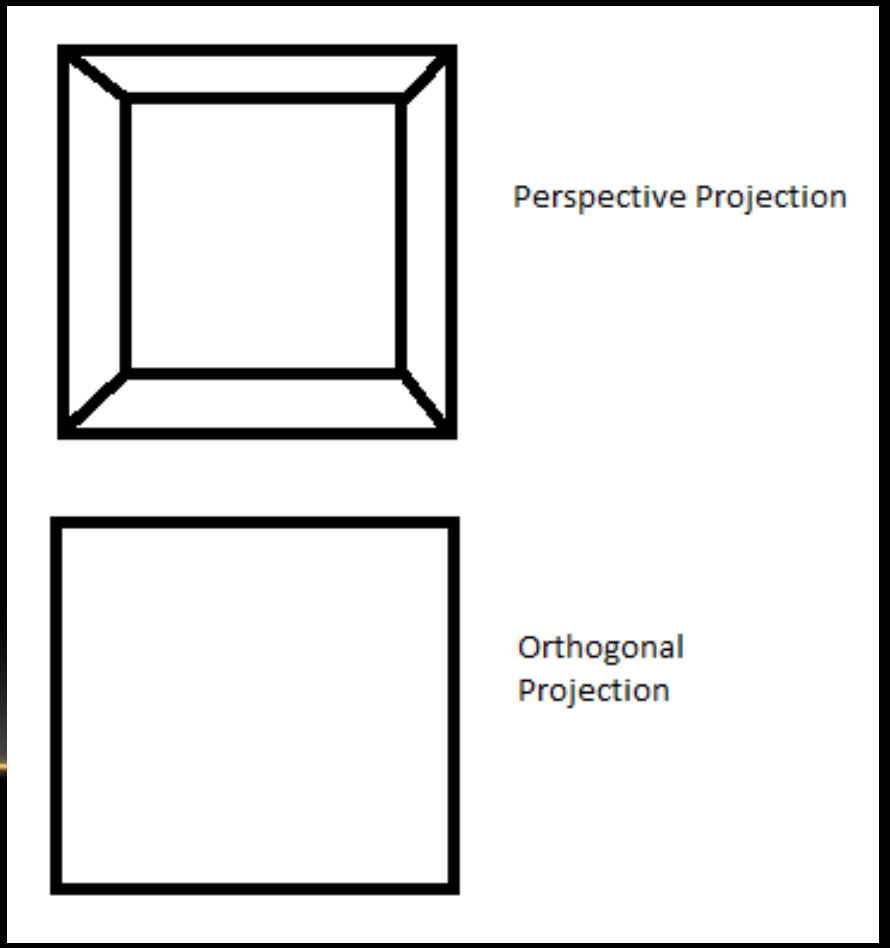
- A heads-up display is a set of indicators used in most games to show the player's current status in a game
  - Status: the score, health and others.
  - Often abbreviated as HUD



CAMERA:
HEADS UP DISPLAY

- How to create a HUD?
- Use Orthogonal Projections!





CAMERA:
HEADS UP DISPLAY

- This method call will start and end our orthogonal projection.

```
// Toggle HUD mode
void SceneText::SetHUD(const bool m_bHUDmode)
{
	if (m_bHUDmode)
	{
	glDisable(GL_DEPTH_TEST);
	Mtx44 ortho;
	ortho.SetToOrtho(0, 80, 0, 60, -10, 10);
	projectionStack.PushMatrix();
		projectionStack.LoadMatrix(ortho);
	}
	else
	{
	projectionStack.PopMatrix();
	glEnable(GL_DEPTH_TEST);
	}
}
```

CAMERA: HEADS UP DISPLAY

- So how do we use these methods?

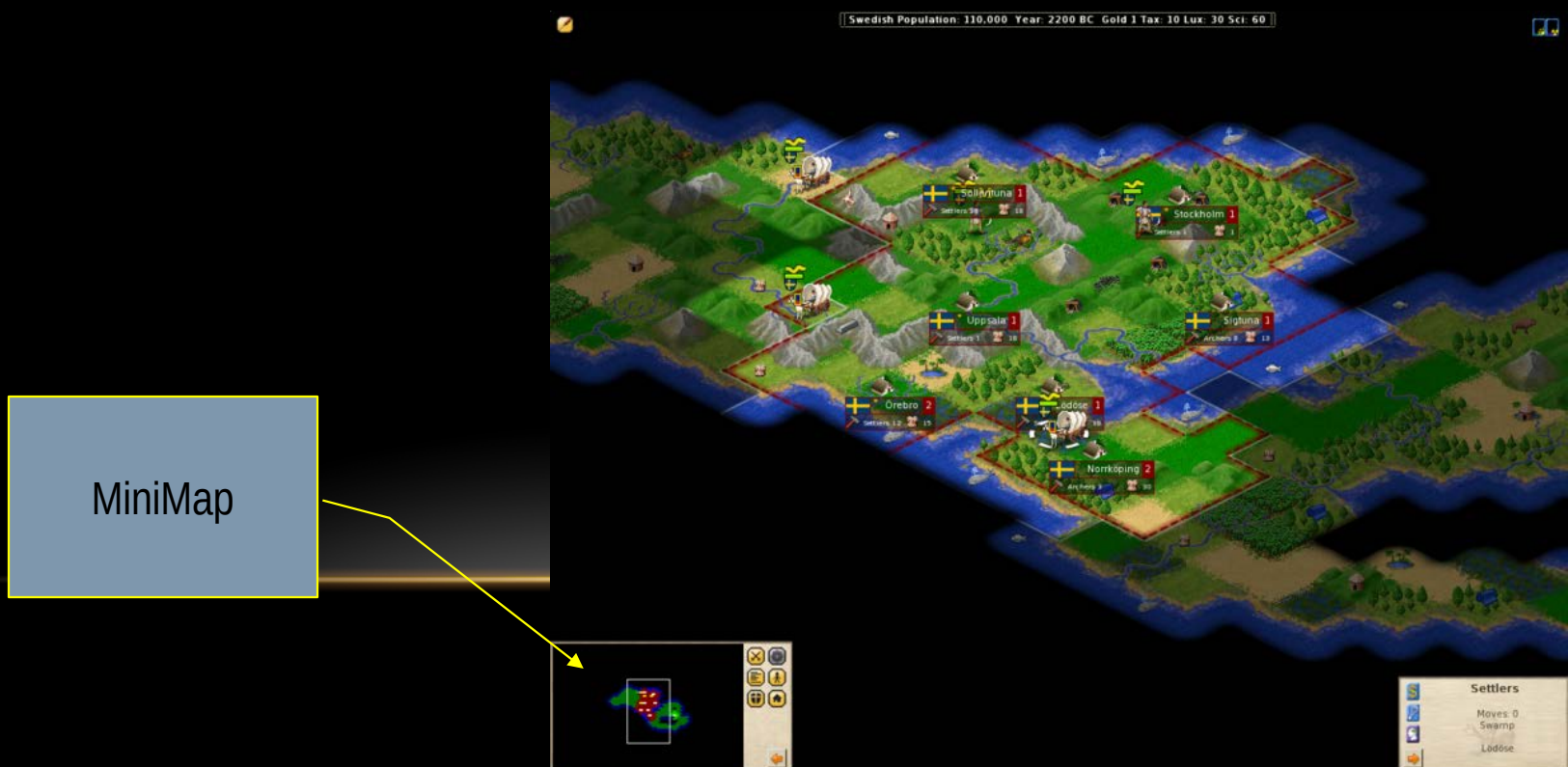
```
//Call our function to start our orthogonal projections
SetHUD(true);

// Draw a text
RenderTextOnScreen(meshList[GEO_TEXT], "Hello Screen", Color(0, 1, 0), 3,
0, 0);

//Now we call our function to end our orthogonal projections
SetHUD(false);
```

CAMERA: MINIMAP

- A mini-map is a miniature map
  - Often placed in a corner of the screen in computer games and video games to aid in reorientation.
  - Usually display traversable terrain, allies, enemies, and important locations or items.



CAMERA: MINIMAP

- How to create a minimap in your game?
  - Draw the terrain as part of the HUD
    - Use 2D Texture Mapping to draw the terrain on a 2D polygon
  - Draw the fixed objects
    - May need to replace with polygons or textures
  - Draw the non-fixed objects
    - May need to replace with polygons or textures
    - Need to show directions?

CAMERA: MINIMAP

Fixed objects

Non-fixed objects with direction

CAMERA: MINIMAP

```
void SceneText::Render(void)
{
  // Render the crosshair
  // Note that Ortho is set to this
  // size ->ortho.SetToOrtho(-80, 80, -60, 60, -10, 10);
  RenderMeshIn2D( m_cMinimap->GetAvatar(), false, 20.0f, 68, -48,
true);
  RenderMeshIn2D( m_cMinimap->GetBorder(), false, 20.0f, 68, -48);
  RenderMeshIn2D( m_cMinimap->GetBackground(), false, 20.0f, 68, -48);
}
```

POINTS TO NOTE ABOUT MINIMAPS

- Should minimaps be circular? Or square?

POINTS TO NOTE ABOUT MINIMAPS

- Should minimaps show the entire map or just part of it?

POINTS TO NOTE ABOUT MINIMAPS

- Should the map rotate or the avatar rotate?

POINTS TO NOTE ABOUT MINIMAPS

- Should the map be 2D or 3D?

POINTS TO NOTE ABOUT MINIMAPS

- How should the avatar and NPCs appear?

SUMMARY

- We have discussed about the main issues with Camera and GUI #2
  - How to create Third-Person Cameras for games such as Socom
    - Use Inertia to reduce motion sickness
    - Techniques to handle camera occlusion with surrounding geometries
  - How to create heads up display using orthogonal projections
  - How to create minimaps to show an overview to the player