CSE 3902: Cameras and Viewports

Justin Holewinski

The Ohio State University

Cameras and Viewports

Dungeons and levels are often larger than what can be shown on the screen.

How do we draw an arbitrary part of the game world to the screen?

Requirements:

- Draw any part of the game world on the screen
- Game objects should not need to know where on the screen they are drawn
- Allow smooth transitions to different viewports

Definitions:

- *Viewport*: A rectangle describing what should be visible
- *Camera*: A software component that manages a viewport

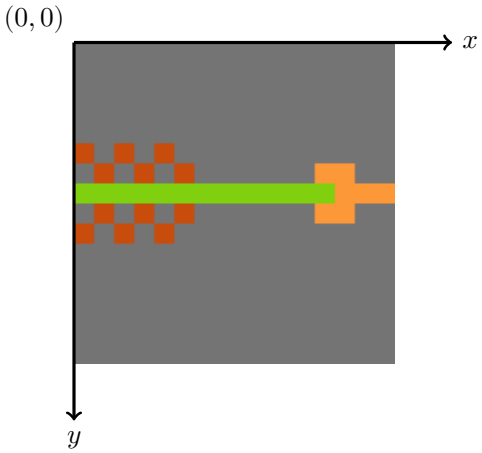
Coordinate Spaces

Local Space: Coordinate space for defining points relative to an object's origin

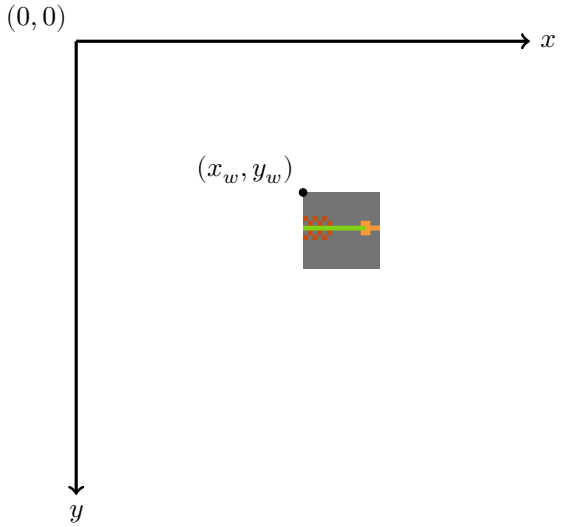
World Space: Coordinate space for defining points in absolute world positions

View Space: Coordinate space for defining points relative to the camera/viewport

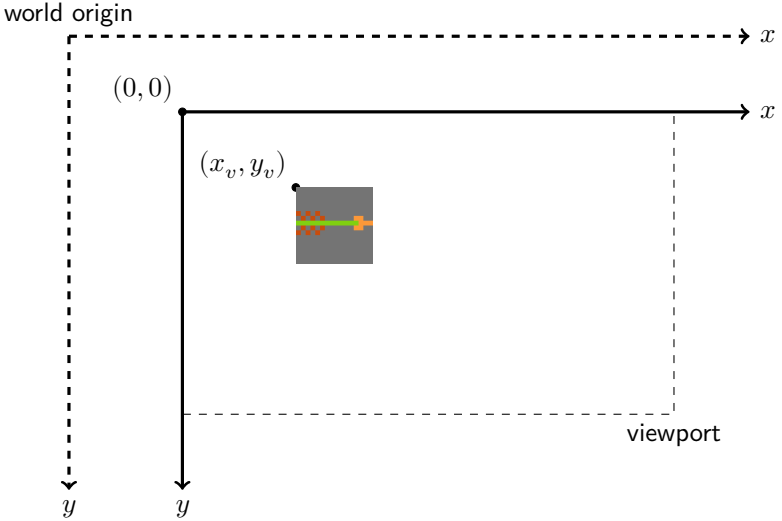
Local Space



World Space



View Space



Coordinate Space Transformation

The above is a simplified view of the standard coordinate transformation pipeline used for 3D graphics.

Model Transform

Goal: convert local coordinates to world coordinates.

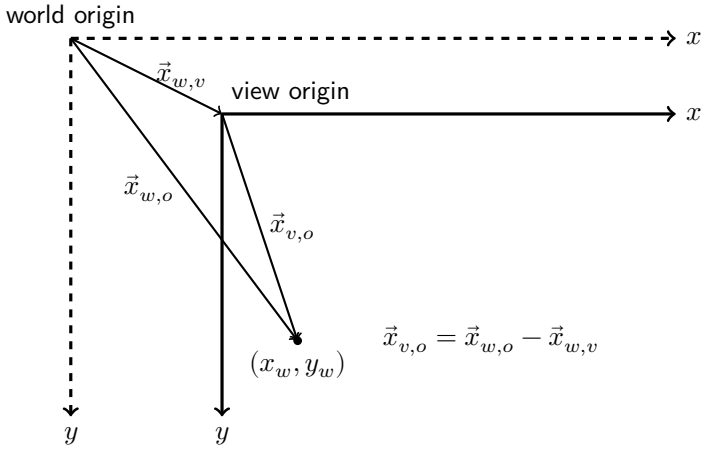
For simple points, just add object's position.

We can assume our sprites are always at $(0, 0)$ in local space, so the world space position is just the object's position.

Consider how you could design your sprite renderer such that an object's position is the *center* of the sprite, not the top-left corner.

View Transform

Goal: convert world coordinates to view coordinates.



Putting It All Together

```
public void DrawGameObject(IGameObject obj, ICamera camera,
                           SpriteBatch batch)
{
    // Compute view origin in world space
    Rectangle viewRect = camera.ViewRectangle;
    Point viewOrigin = new Point2(viewRect.X, viewRect.Y);

    // Get sprite position in world space
    Point spritePosWorld = obj.Position;

    // Compute sprite position in view space
    Point spritePosView = spritePosWorld - viewOrigin;

    obj.DrawAt(spritePosView, batch);
}
```

The above shows an example of computing sprite drawing positions, but is not necessarily a good example of good software quality.