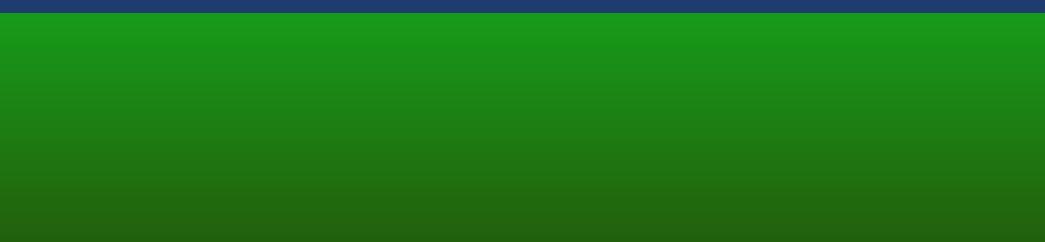
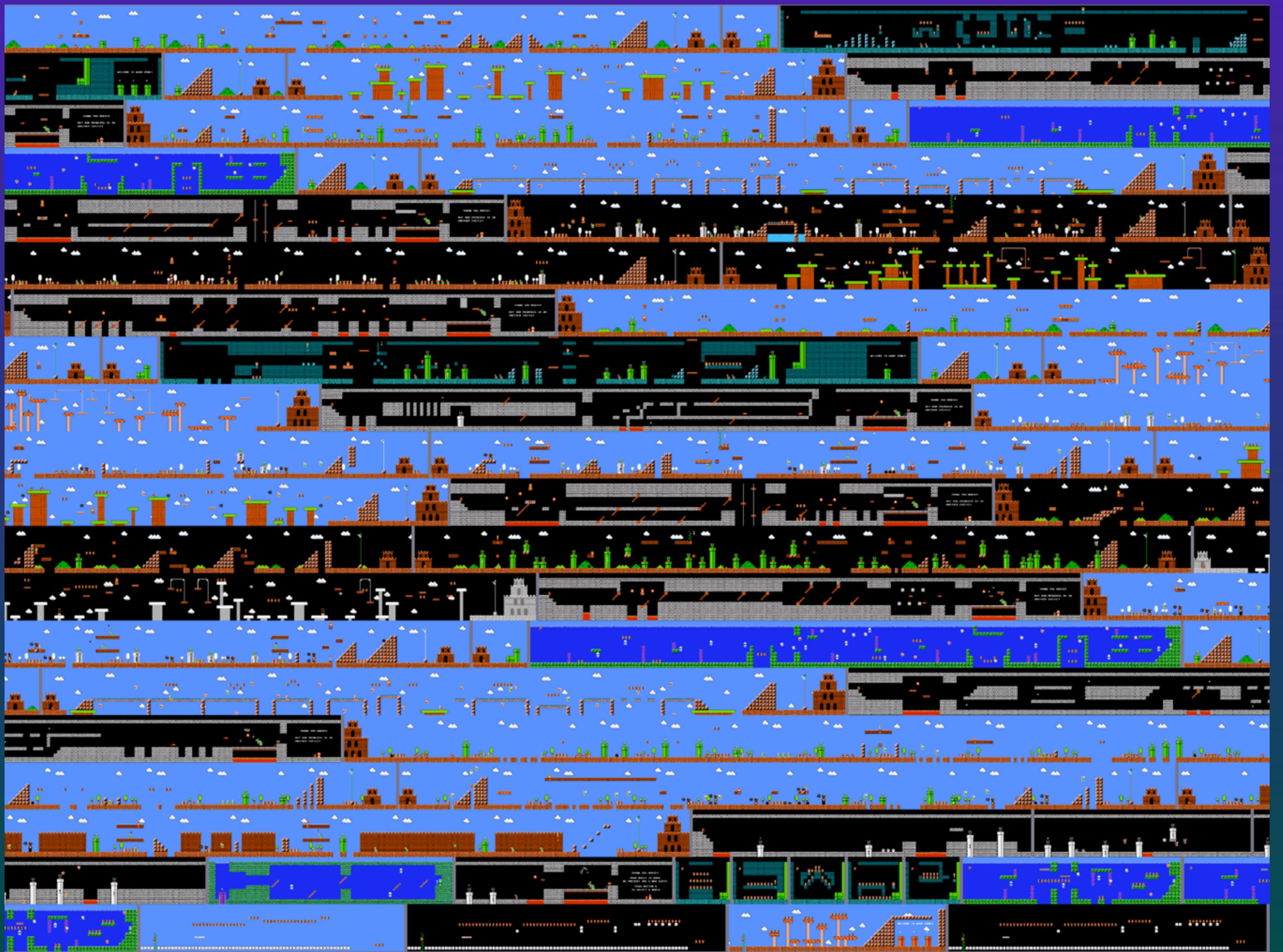


# Creating worlds.

## Part 1









HEALTH



FUEL



BLAST-CARBINE

FPS: 59

TEXTURE: (PRESS N TO SWITCH)

TILED: FALSEPRESS T TO TOGGLE

X: 1828.409 - Y: 1476.546 - ROTATED: 3.150086

COLOR (KEYS UIOP TO CHANGE) --- RED: 1 - GREEN: 1 - BLUE: 1 - ALPHA: 1

MASS: IPRESS +/- TO INCREASE/DECREASE

MOVEABLE: OPRESS M TO TOGGLE

NO ROTATE: FALSEPRESS R TO TOGGLE

GEM FROM TEXTURE: FALSEPRESS Z TO TOGGLE

HURT: D - PUSH (A/S): D

COLLISION TYPE:



# Tile-based levels.



MARIO  
000500

0x01

WORLD  
1-1

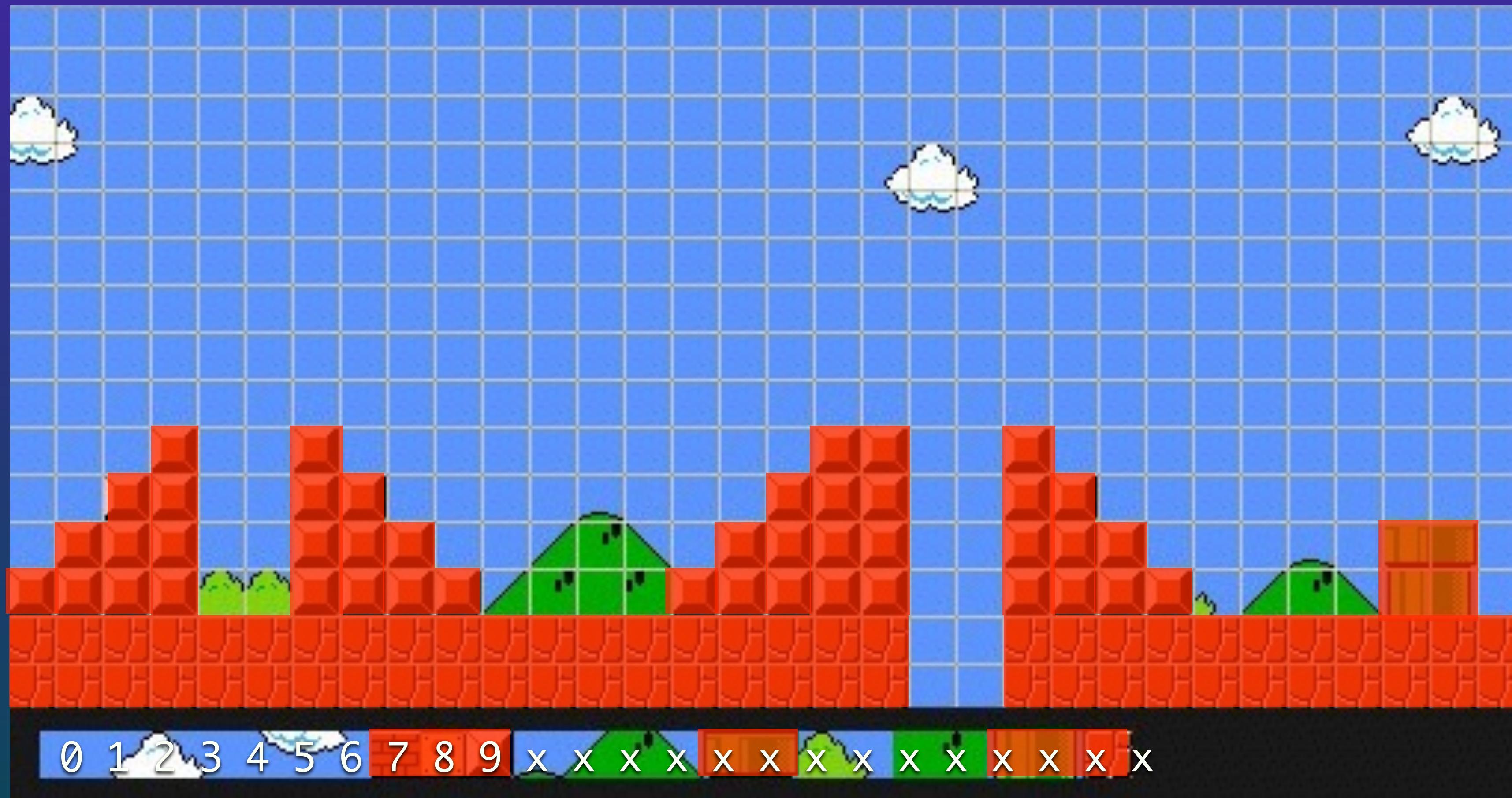
TIME  
321











`solids = 7,8,9,14,15,20,21,22`





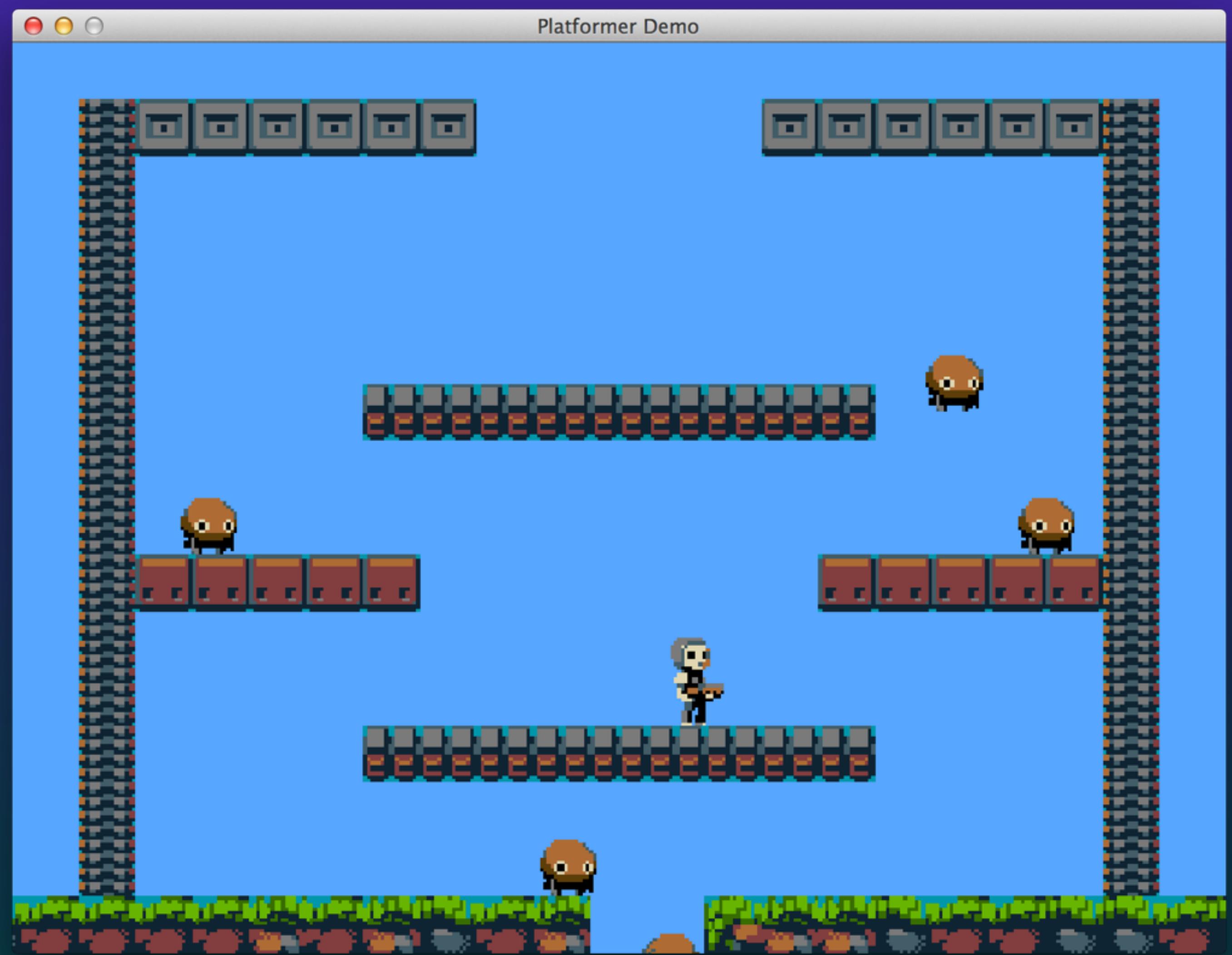
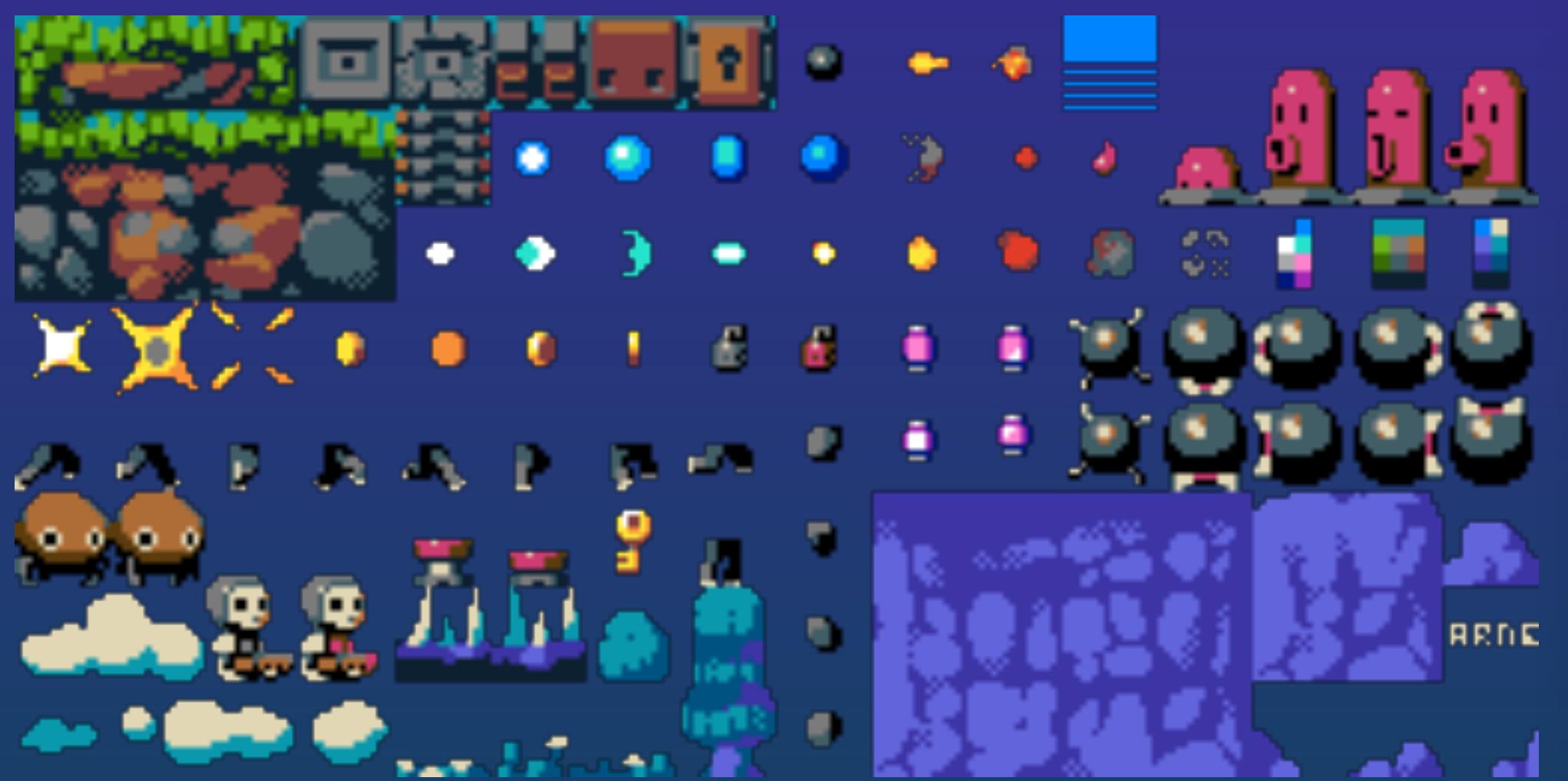


4

4

\$0





```
void ClassDemoApp::buildLevel() {
    int blockIndex = 0;

    for(int i=0; i < 9; i++) {
        blocks[blockIndex].sprite = SheetSprite(spriteSheetTexture, 5);
        blocks[blockIndex].width = 2.0/8.0f;
        blocks[blockIndex].height = 2.0/8.0f;
        blocks[blockIndex].isStatic = true;
        blocks[blockIndex].y = (2.0/8.0f * -4) - 1.0f/8.0f;
        blocks[blockIndex].x = (-2.0/8.0f * 4) + (2.0/8.0f * i);
        entities.push_back(&blocks[blockIndex]);
        blockIndex++;
    }

    for(int i=0; i < 9; i++) {
        blocks[blockIndex].sprite = SheetSprite(spriteSheetTexture, 5);
        blocks[blockIndex].width = 2.0/8.0f;
        blocks[blockIndex].height = 2.0/8.0f;
        blocks[blockIndex].isStatic = true;
        blocks[blockIndex].y = (2.0/8.0f * 2) - 1.0f/8.0f;
        blocks[blockIndex].x = (-2.0/8.0f * 4) + (2.0/8.0f * i);
        entities.push_back(&blocks[blockIndex]);
        blockIndex++;
    }

    for(int i=0; i < 5; i++) {
        blocks[blockIndex].sprite = SheetSprite(spriteSheetTexture, 6);
        blocks[blockIndex].width = 2.0/8.0f;
        blocks[blockIndex].height = 2.0/8.0f;
        blocks[blockIndex].isStatic = true;
        blocks[blockIndex].y = (2.0/8.0f * -1) - 1.0f/8.0f;
        blocks[blockIndex].x = (-2.0/8.0f * 8) + (2.0/8.0f * i);
        entities.push_back(&blocks[blockIndex]);
    }
}
```

```
void ClassDemoApp::buildLevel() {
    int blockIndex = 0;

    for(int i=0; i < 9; i++) {
        blocks[blockIndex].sprite = SheetSprite(spriteSheetTexture, 5);
        blocks[blockIndex].width = 2.0/8.0f;
        blocks[blockIndex].height = 2.0/8.0f;
        blocks[blockIndex].isStatic = true;
        blocks[blockIndex].y = (2.0/8.0f * -4) - 1.0f/8.0f;
        blocks[blockIndex].x = (-2.0/8.0f * 4) + (2.0/8.0f * i);
        entities.push_back(&blocks[blockIndex]);
        blockIndex++;
    }

    for(int i=0; i < 9; i++) {
        blocks[blockIndex].sprite = SheetSprite(spriteSheetTexture, 5);
        blocks[blockIndex].width = 2.0/8.0f;
        blocks[blockIndex].height = 2.0/8.0f;
        blocks[blockIndex].isStatic = false;
        blocks[blockIndex].y = (2.0/8.0f * 2) - 1.0f/8.0f;
        blocks[blockIndex].x = (-2.0/8.0f * 4) + (2.0/8.0f * i);
        entities.push_back(&blocks[blockIndex]);
        blockIndex++;
    }

    for(int i=0, i < 5; i++) {
        blocks[blockIndex].sprite = SheetSprite(spriteSheetTexture, 6);
        blocks[blockIndex].width = 2.0/8.0f;
        blocks[blockIndex].height = 2.0/8.0f;
```

# Defining a tile map level.



Storing level data as a 2-dimensional array.

```
unsigned char levelData[LEVEL_HEIGHT][LEVEL_WIDTH];
```



```
    unsigned char levelData[0] = {0,20,4,4,4,4,4,4,0,0,0,0,0,4,4,4,4,4,4,20,0};
```

```
unsigned char level1Data[LEVEL_HEIGHT][LEVEL_WIDTH] =  
{  
    {11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11},  
    {0,20,4,4,4,4,4,4,0,0,0,0,0,0,4,4,4,4,4,4,4,4,20,0},  
    {0,20,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,20,0},  
    {0,20,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,20,0},  
    {0,20,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,20,0},  
    {0,20,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,20,0},  
    {0,20,0,0,0,0,0,6,6,6,6,6,6,6,6,0,0,0,0,0,0,0,20,0},  
    {0,20,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,20,0},  
    {0,20,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,20,0},  
    {0,20,6,6,6,6,6,0,0,0,0,0,0,0,0,6,6,6,6,6,6,20,0},  
    {0,20,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,20,0},  
    {0,20,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,20,0},  
    {0,20,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,20,0},  
    {0,20,125,118,0,0,116,0,0,0,0,0,0,0,0,0,0,0,0,117,0,127,20,0},  
    {2,2,2,2,2,2,2,2,3,0,0,1,2,2,2,2,2,2,2,2,2,2},  
    {32,33,33,34,32,33,33,34,33,35,100,101,35,32,33,32,34,32,33,32,33}  
};
```

```
void ClassDemoApp::buildLevel() {  
    memcpy(levelData, level1Data, LEVEL_HEIGHT*LEVEL_WIDTH);  
}
```

# Rendering a tile map level.

Need to render the entire tilemap in one array.

H e l l o

Go through the tiles line by line and add vertices to a single array. Keep in mind that our Y axis points upwards, while tile indexes count downwards.

```
for(int y=0; y < LEVEL_HEIGHT; y++) {
    for(int x=0; x < LEVEL_WIDTH; x++) {

        float u = (float)((int)levelData[y][x]) % SPRITE_COUNT_X / (float) SPRITE_COUNT_X;
        float v = (float)((int)levelData[y][x]) / SPRITE_COUNT_X / (float) SPRITE_COUNT_Y;

        spriteWidth = 1.0f/(float)SPRITE_COUNT_X;
        spriteHeight = 1.0f/(float)SPRITE_COUNT_Y;

        vertexData.insert(vertexData.end(), {
            TILE_SIZE * x, -TILE_SIZE * y,
            TILE_SIZE * x, (-TILE_SIZE * y)-TILE_SIZE,
            (TILE_SIZE * x)+TILE_SIZE, (-TILE_SIZE * y)-TILE_SIZE,
            (TILE_SIZE * x)+TILE_SIZE, -TILE_SIZE * y
        });

        texCoordData.insert(texCoordData.end(), { u, v,
            u, v+(spriteHeight),
            u+spriteWidth, v+(spriteHeight),
            u+spriteWidth, v
        });
    }
}
```

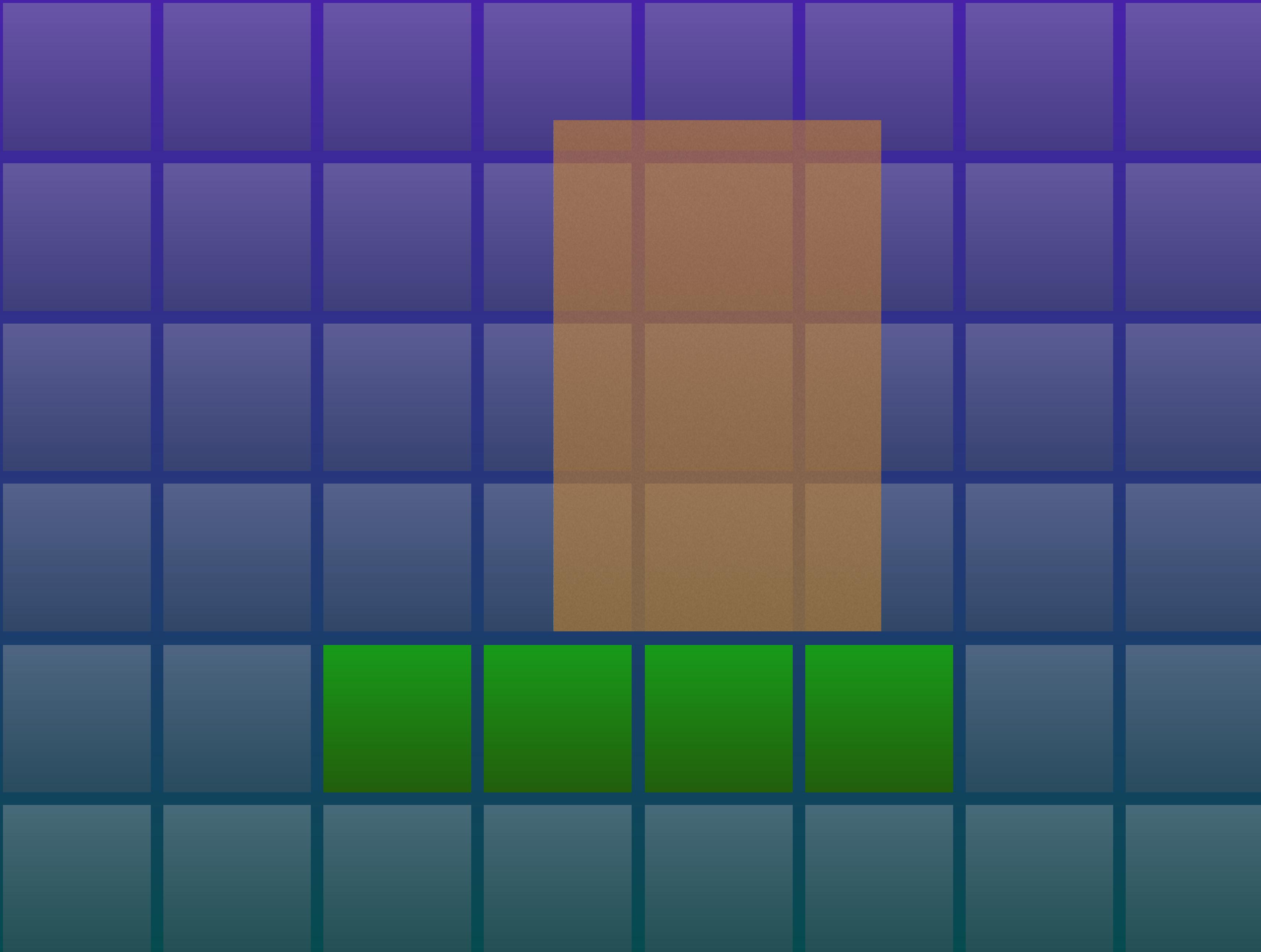
# We can optimize by not drawing empty grid tiles at all.

```
for(int y=0; y < LEVEL_HEIGHT; y++) {  
    for(int x=0; x < LEVEL_WIDTH; x++) {  
  
        if(levelData[y][x] != 0) {  
            // add vertices  
        }  
    }  
}
```

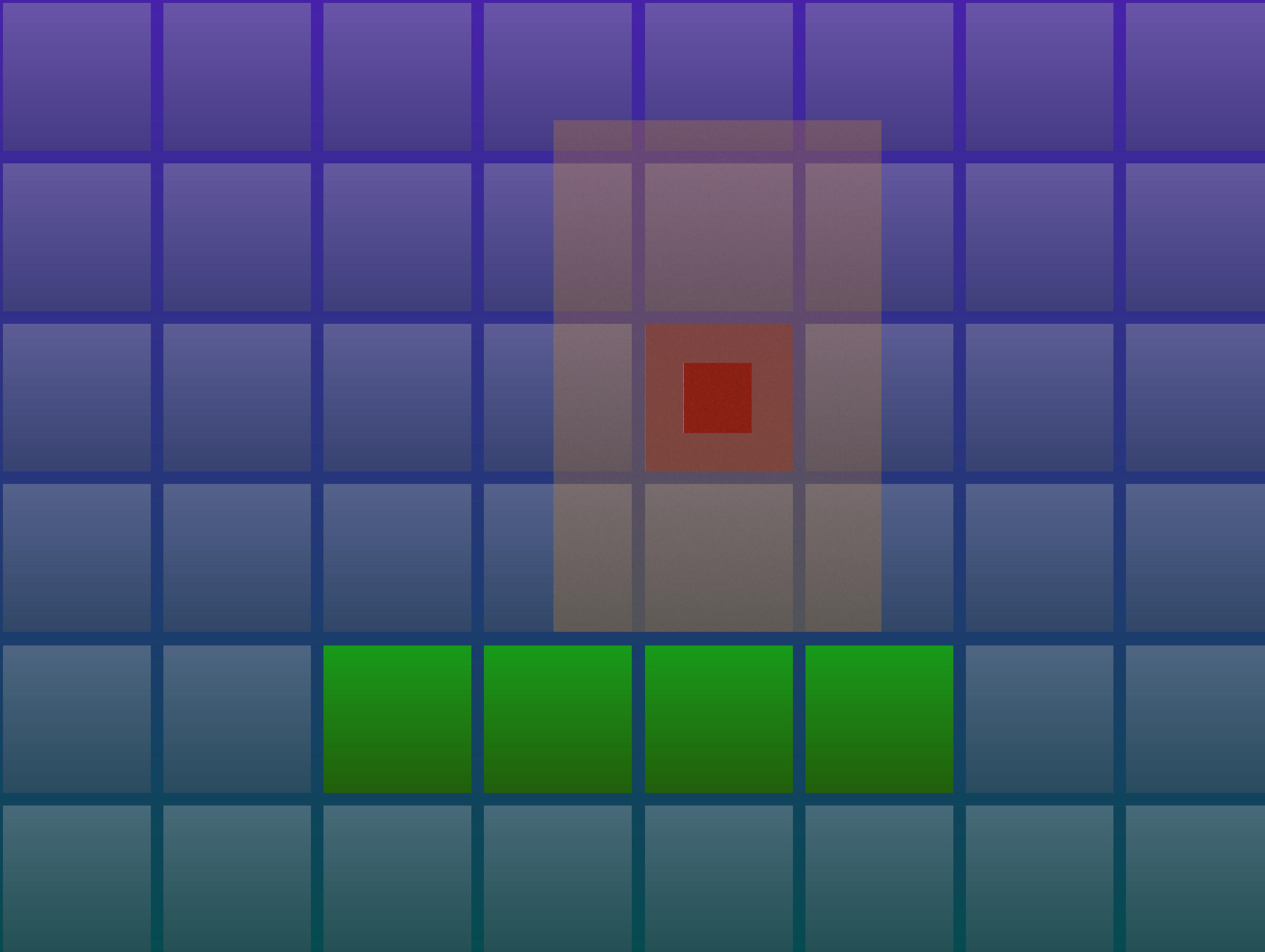
# Translate to center on screen.

```
glLoadIdentity();
glTranslatef(-TILE_SIZE * LEVEL_WIDTH/2, TILE_SIZE * LEVEL_HEIGHT/2, 0.0f);
```

# Colliding with a tilemap.



Convert our entity position  
to the grid coordinates and check  
if that tile is solid.

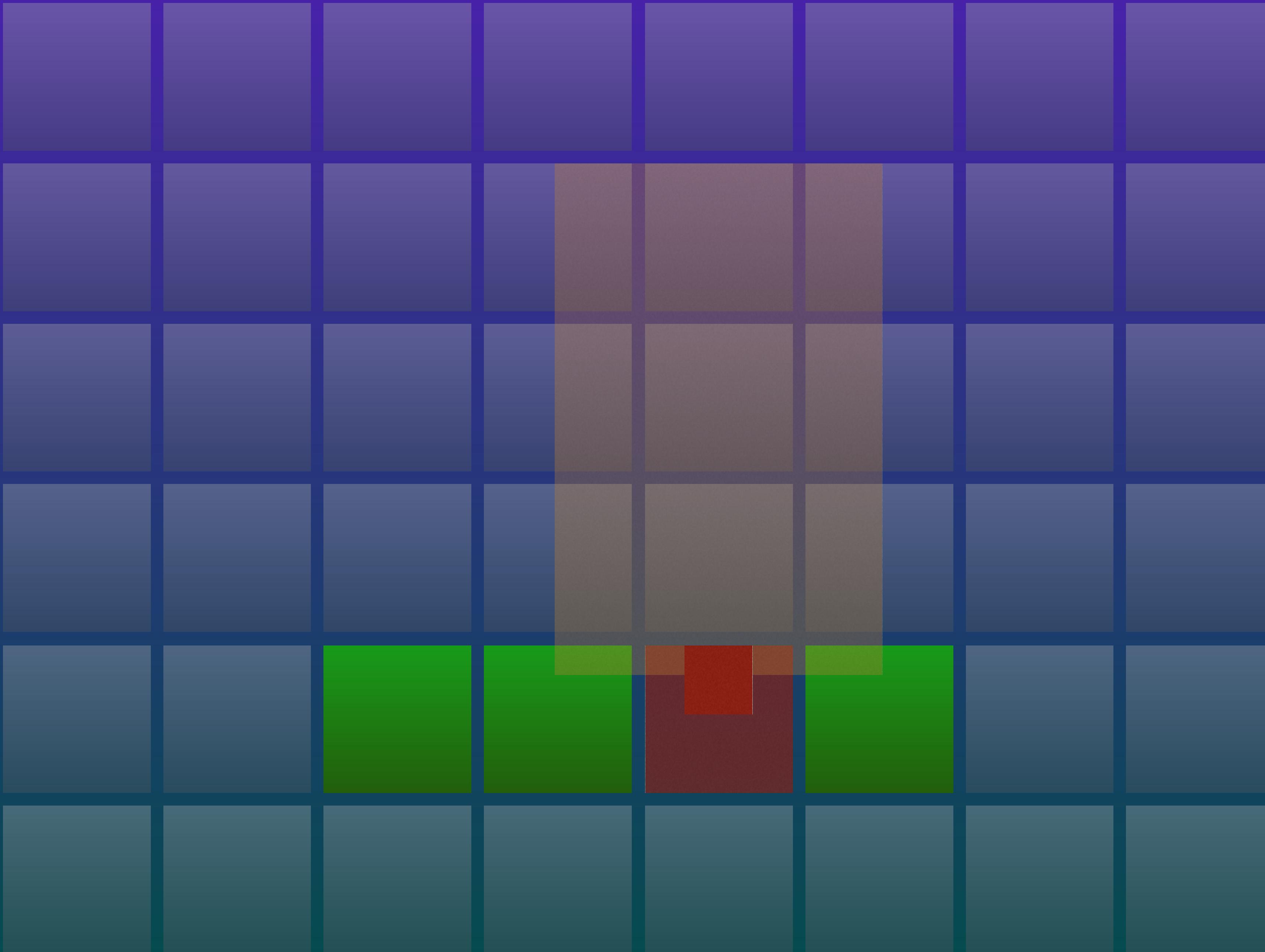


```
void worldToTileCoordinates(float worldX, float worldY, int *gridX, int *gridY) {  
    *gridX = (int)(worldX / TILE_SIZE);  
    *gridY = (int)(-worldY / TILE_SIZE);  
}
```

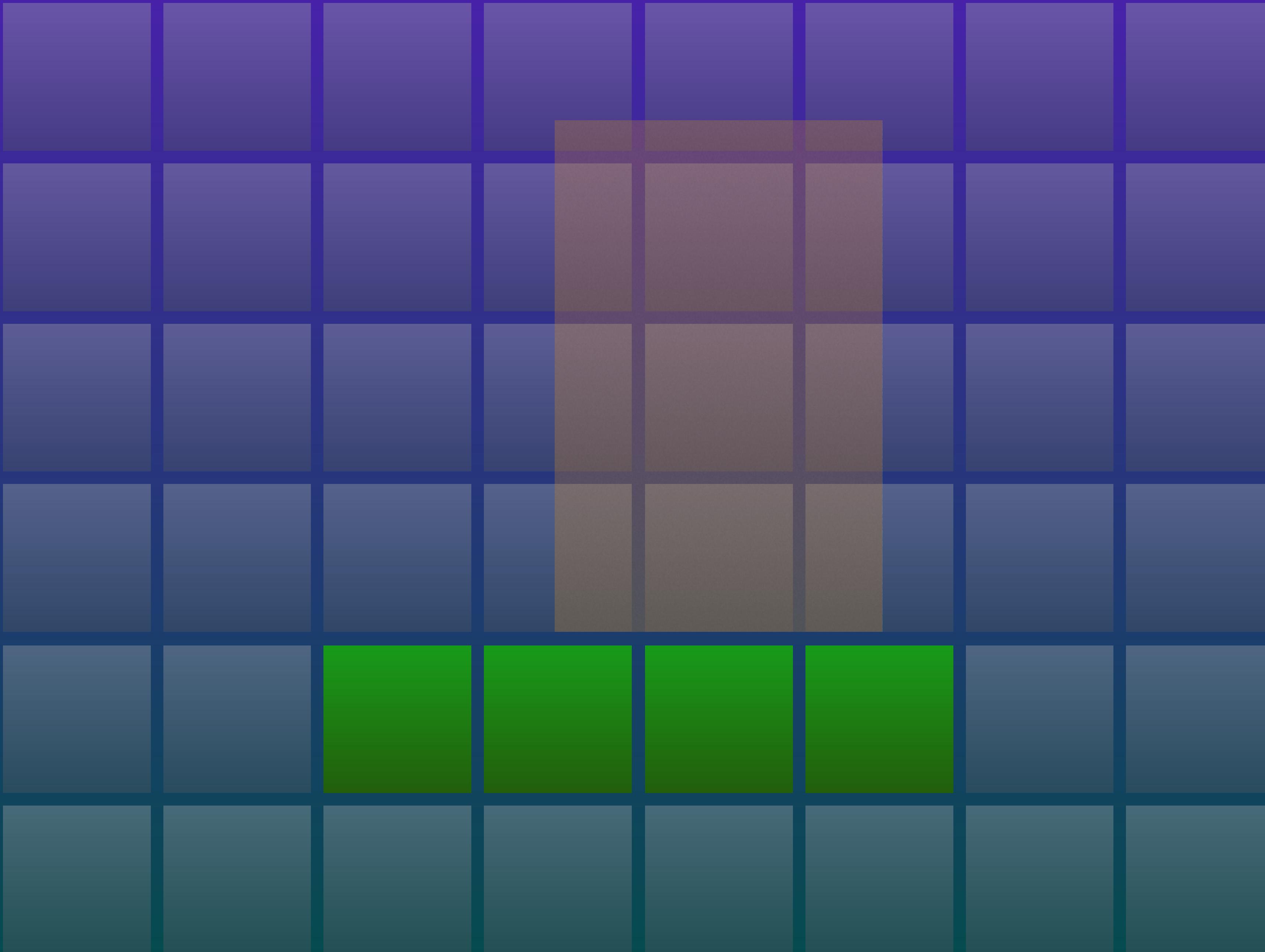
If we are shifting our tilemap rendering to be centered on the screen, we need to account for that offset.

```
void worldToTileCoordinates(float worldX, float worldY, int *gridX, int *gridY) {
    *gridX = (int)((worldX + (WORLD_OFFSET_X)) / TILE_SIZE);
    *gridY = (int)((-worldY + (WORLD_OFFSET_Y)) / TILE_SIZE);
}
```

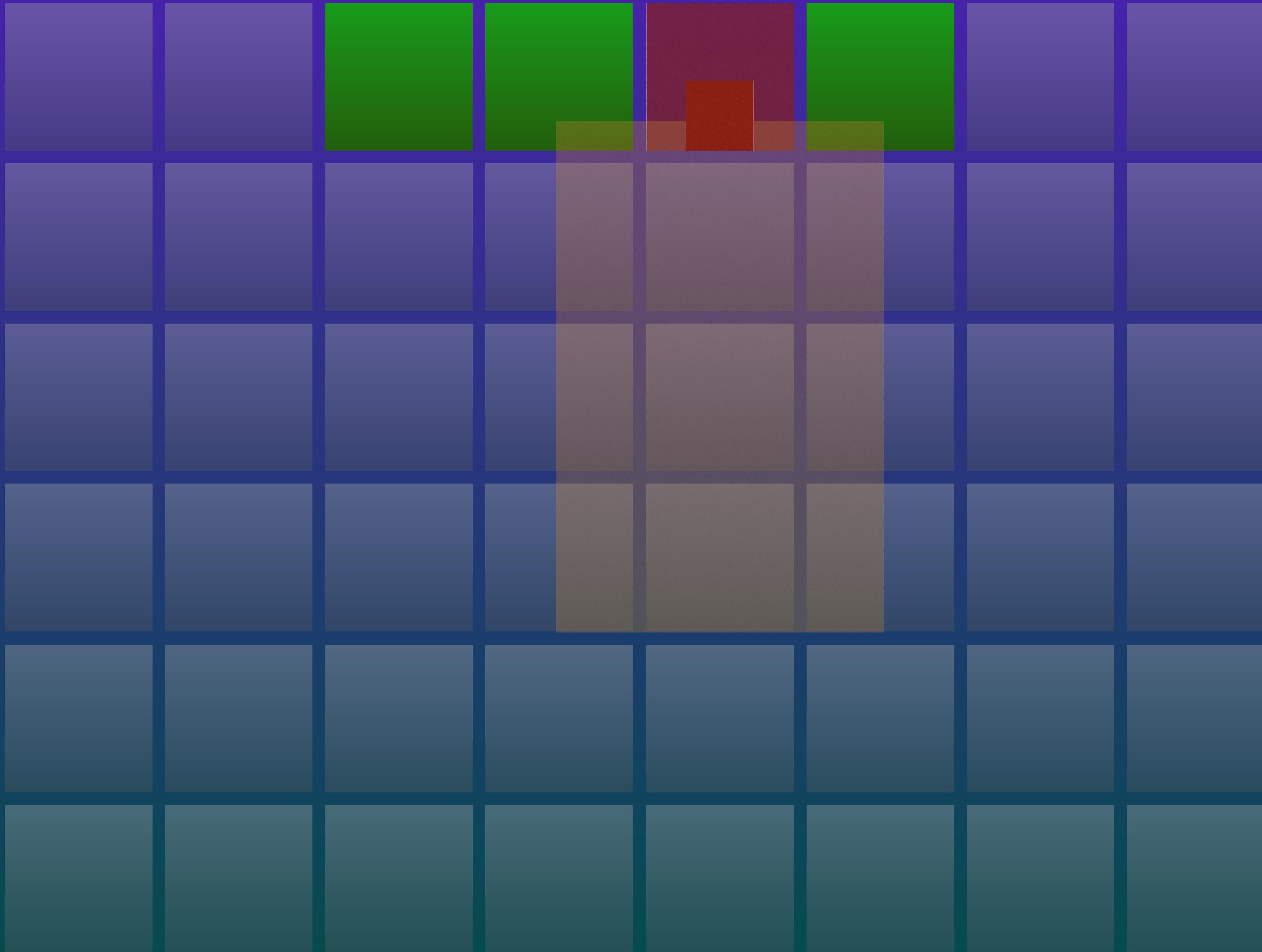
We don't want to collide using the entity's center, so check the point at the bottom (-half height).



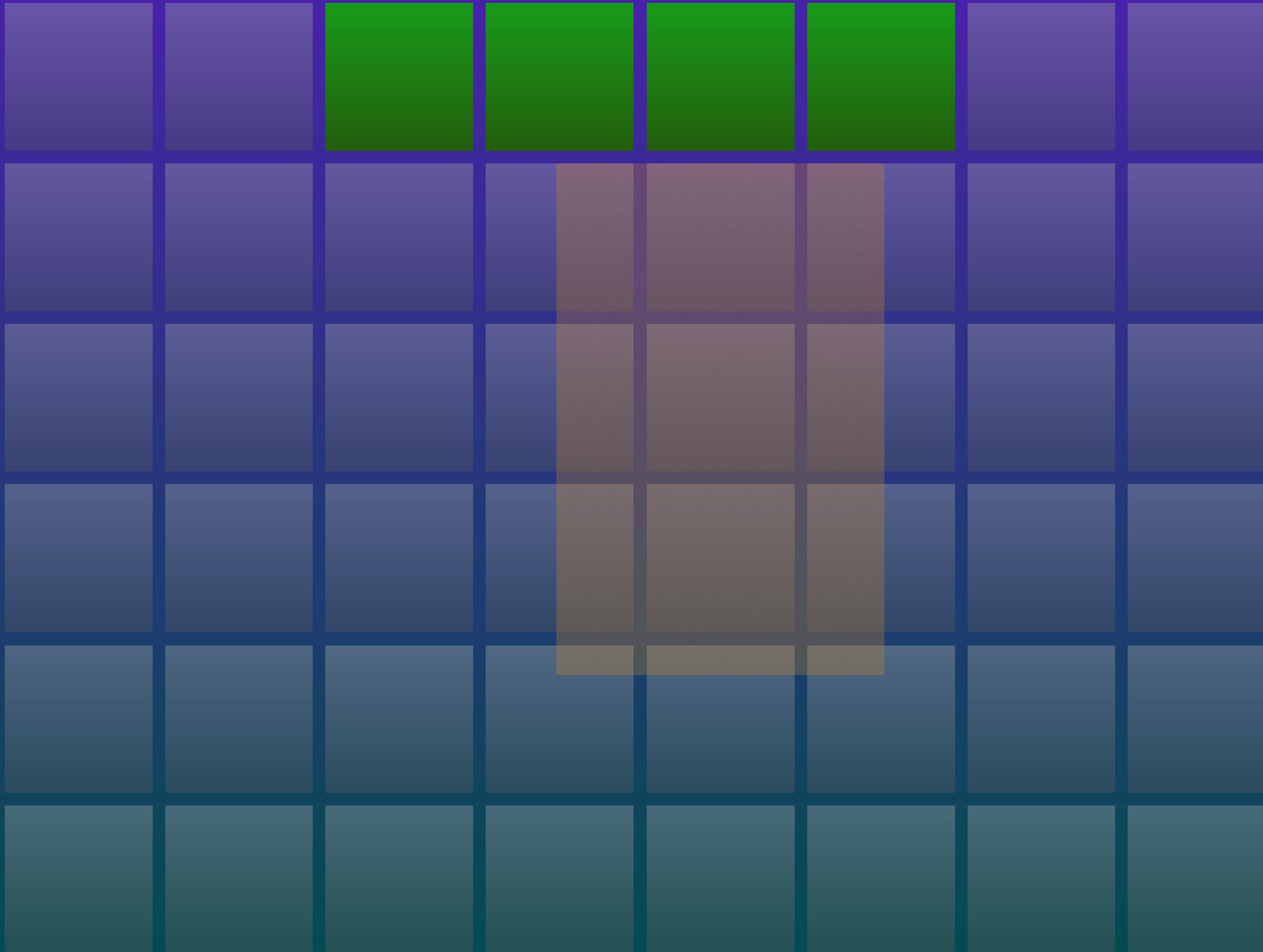
If the tile is solid, adjust our Y-coordinate up by the difference between the point we are testing and the top of the tile and set Y-velocity to 0, set bottom collision flag..



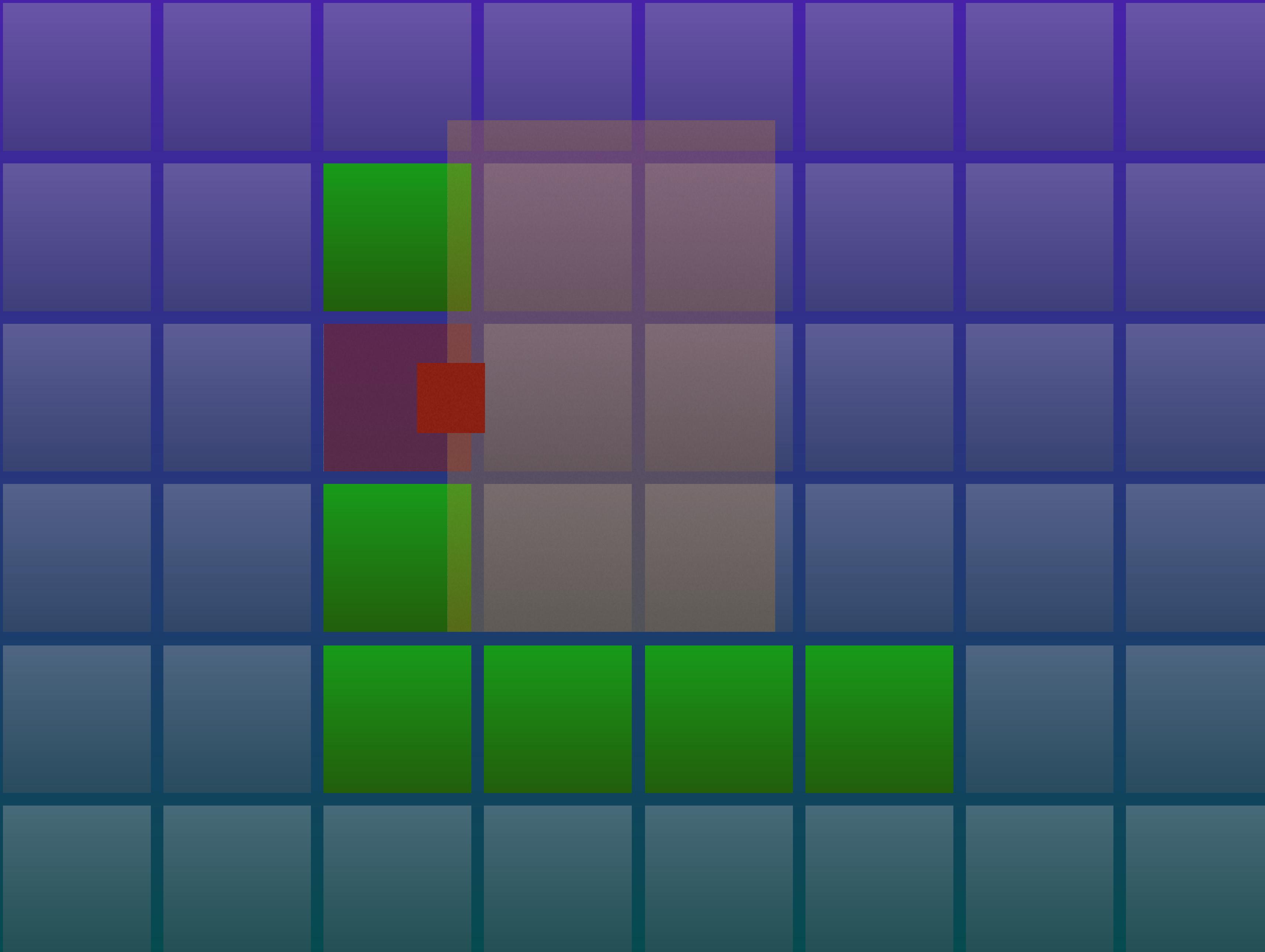
Now, check the point at the top...



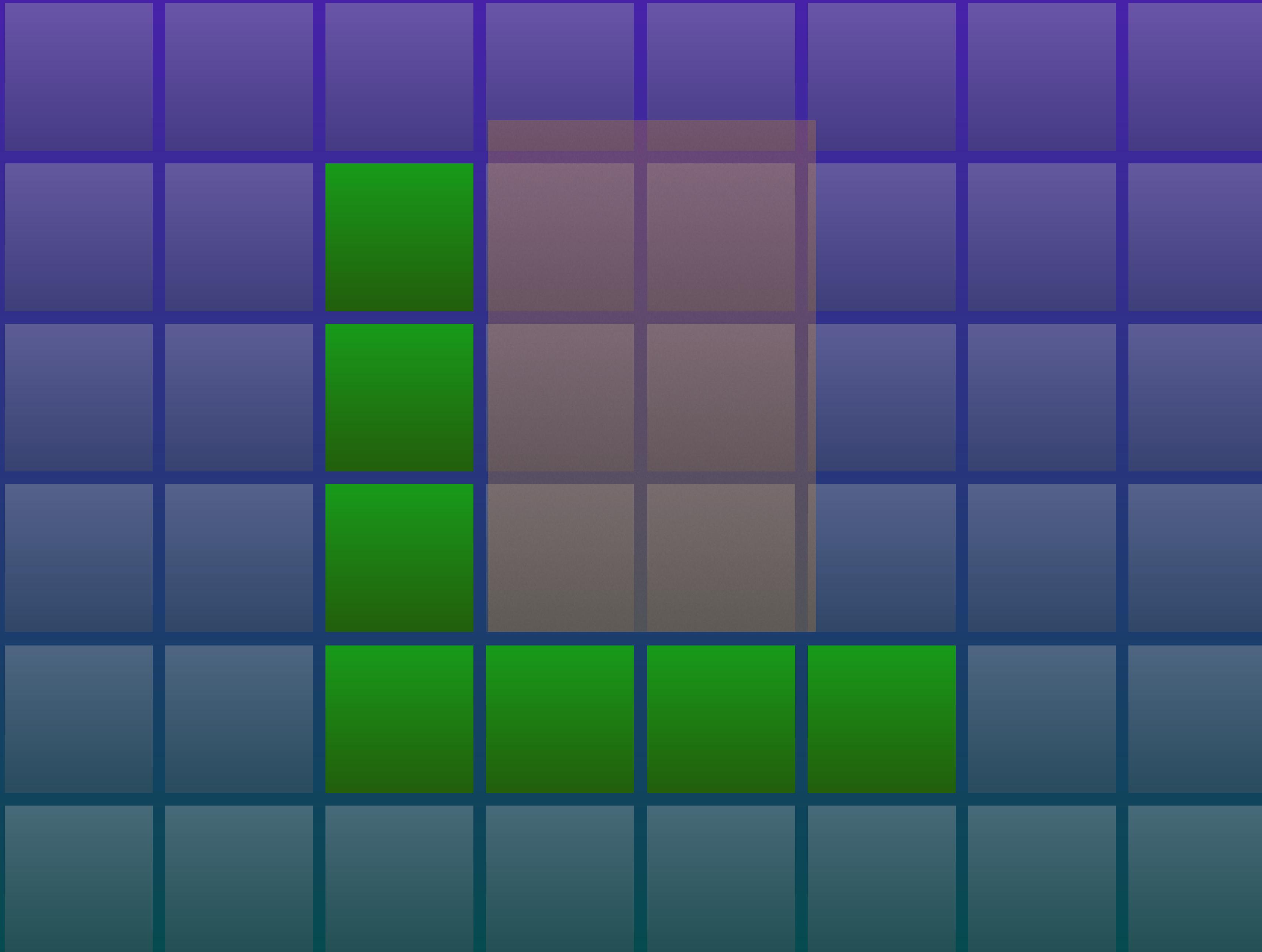
If the tile is solid, adjust our Y-coordinate down by the difference between the point we are testing and the bottom of the tile and set Y-velocity to 0, set top collision flag.



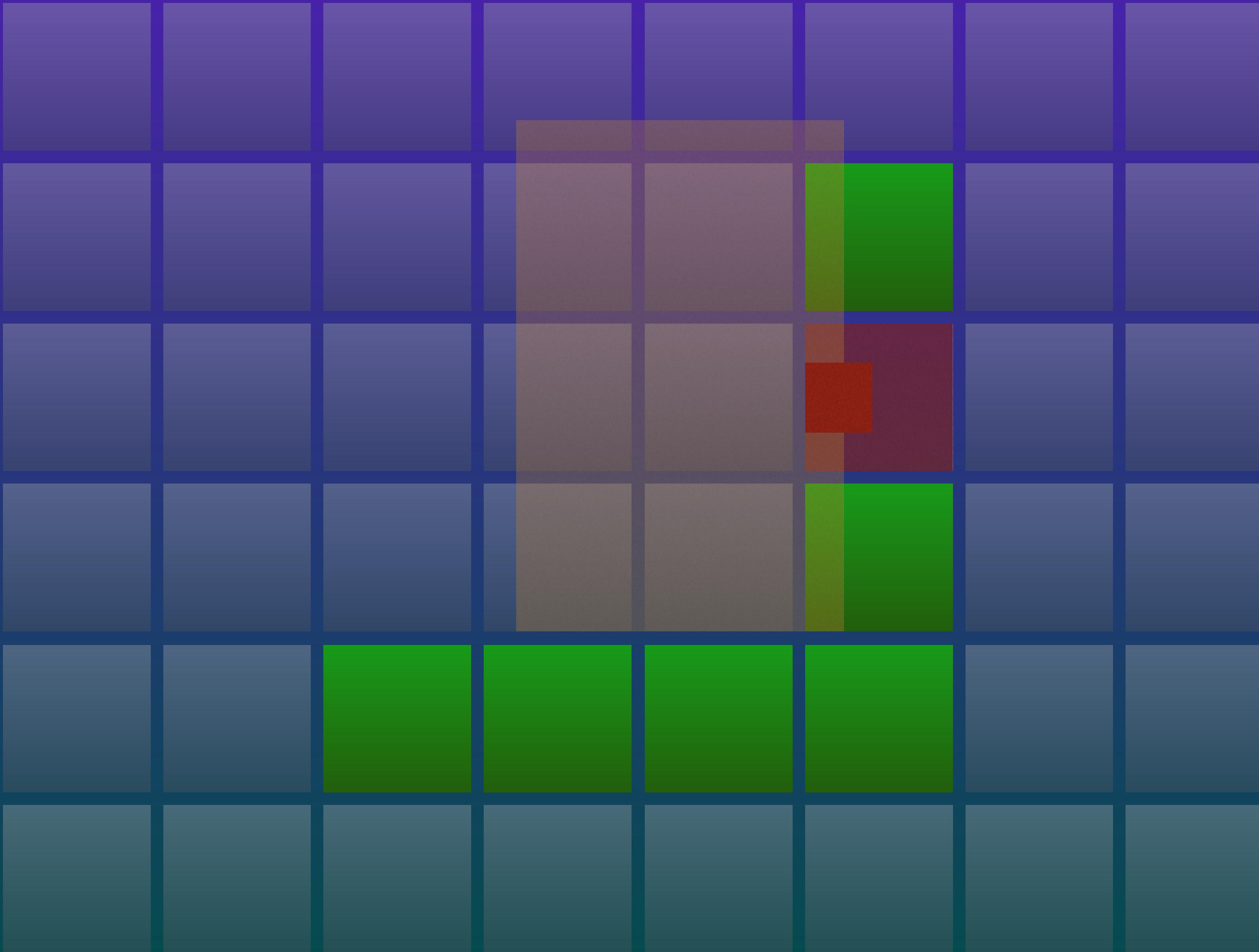
Now, on the left...



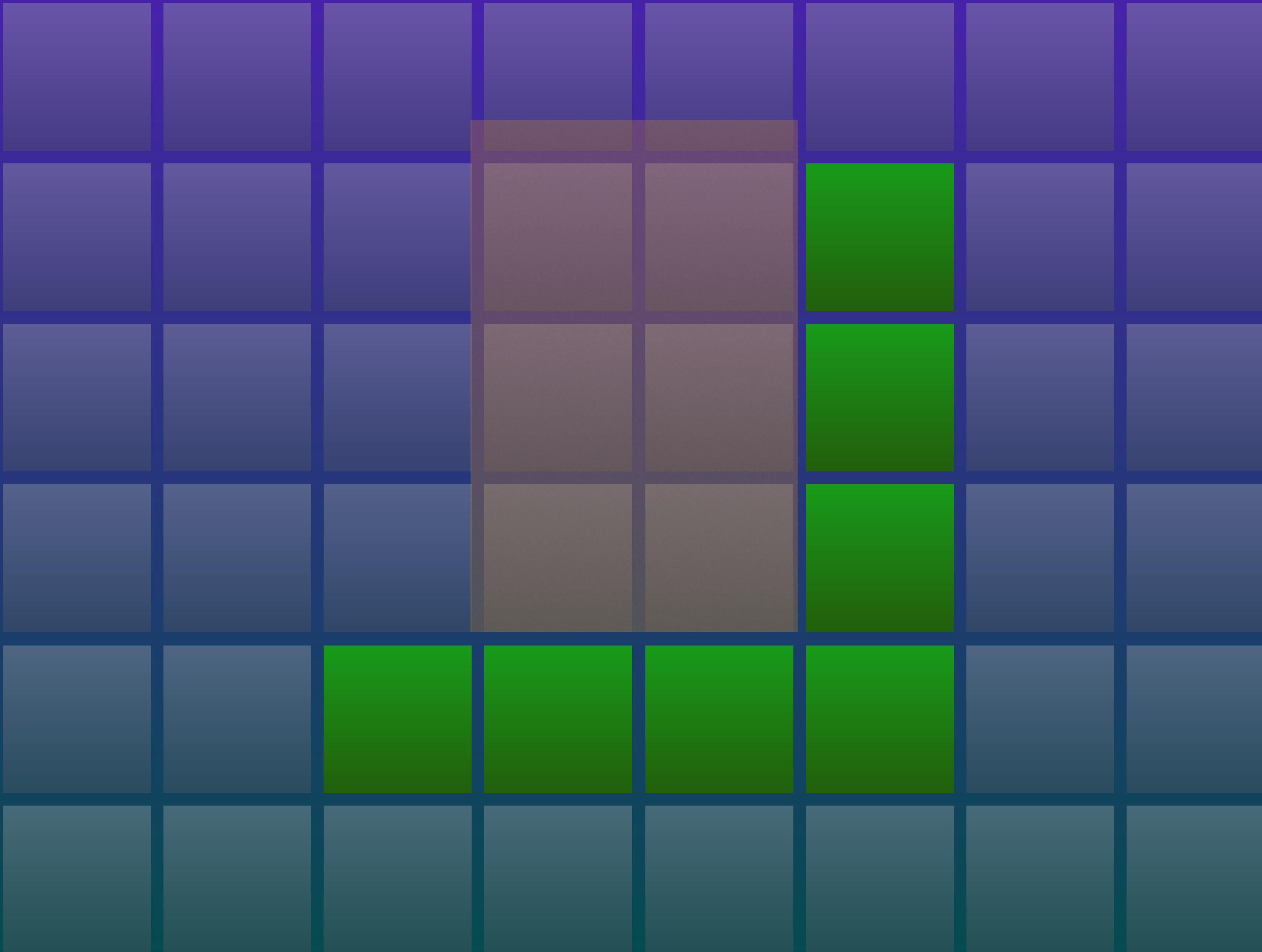
If the tile is solid, adjust our X-coordinate right by the difference between the point we are testing and the right of the tile and set X-velocity to 0, set left collision flag.



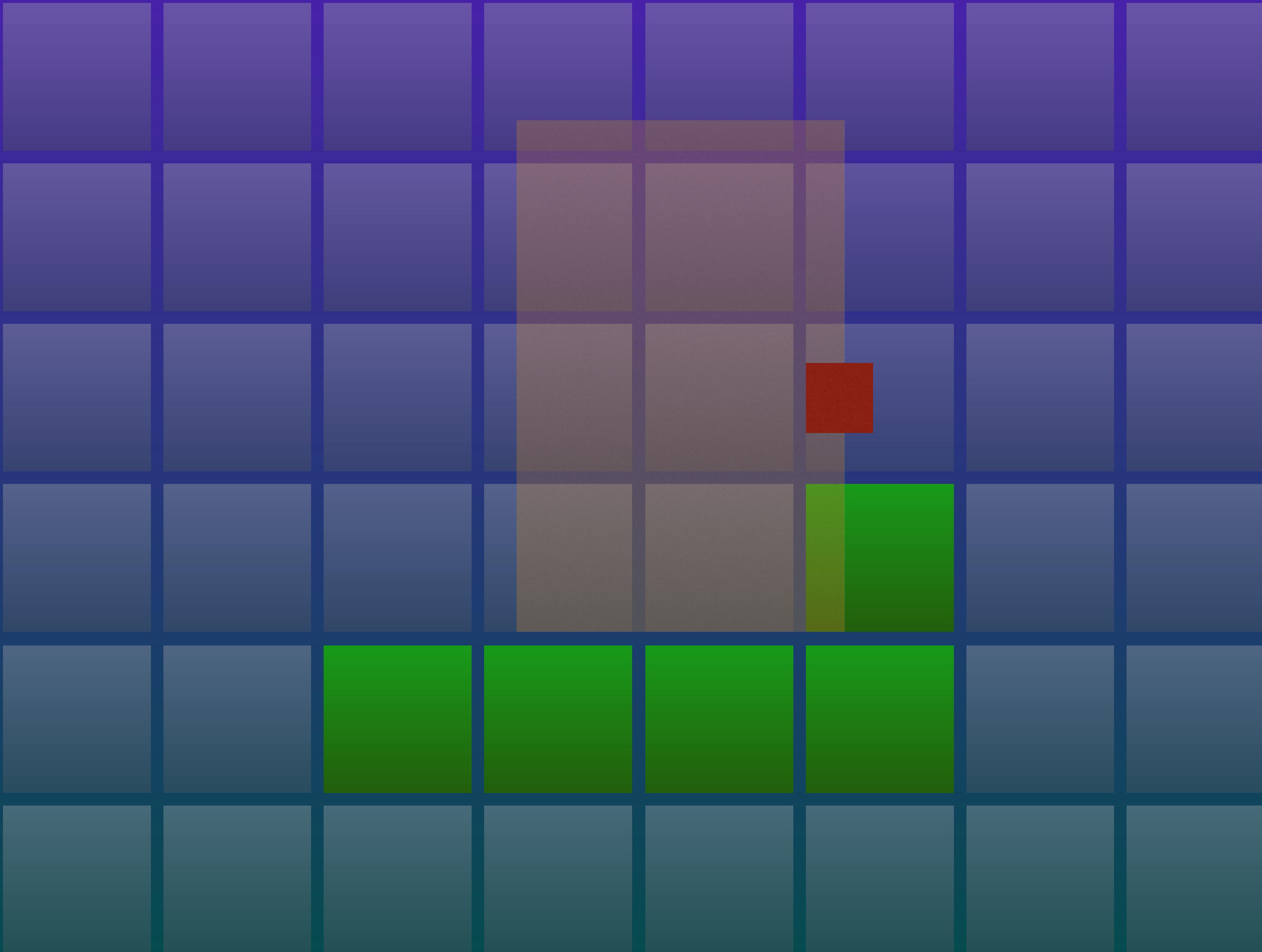
Now, finally, on the right...

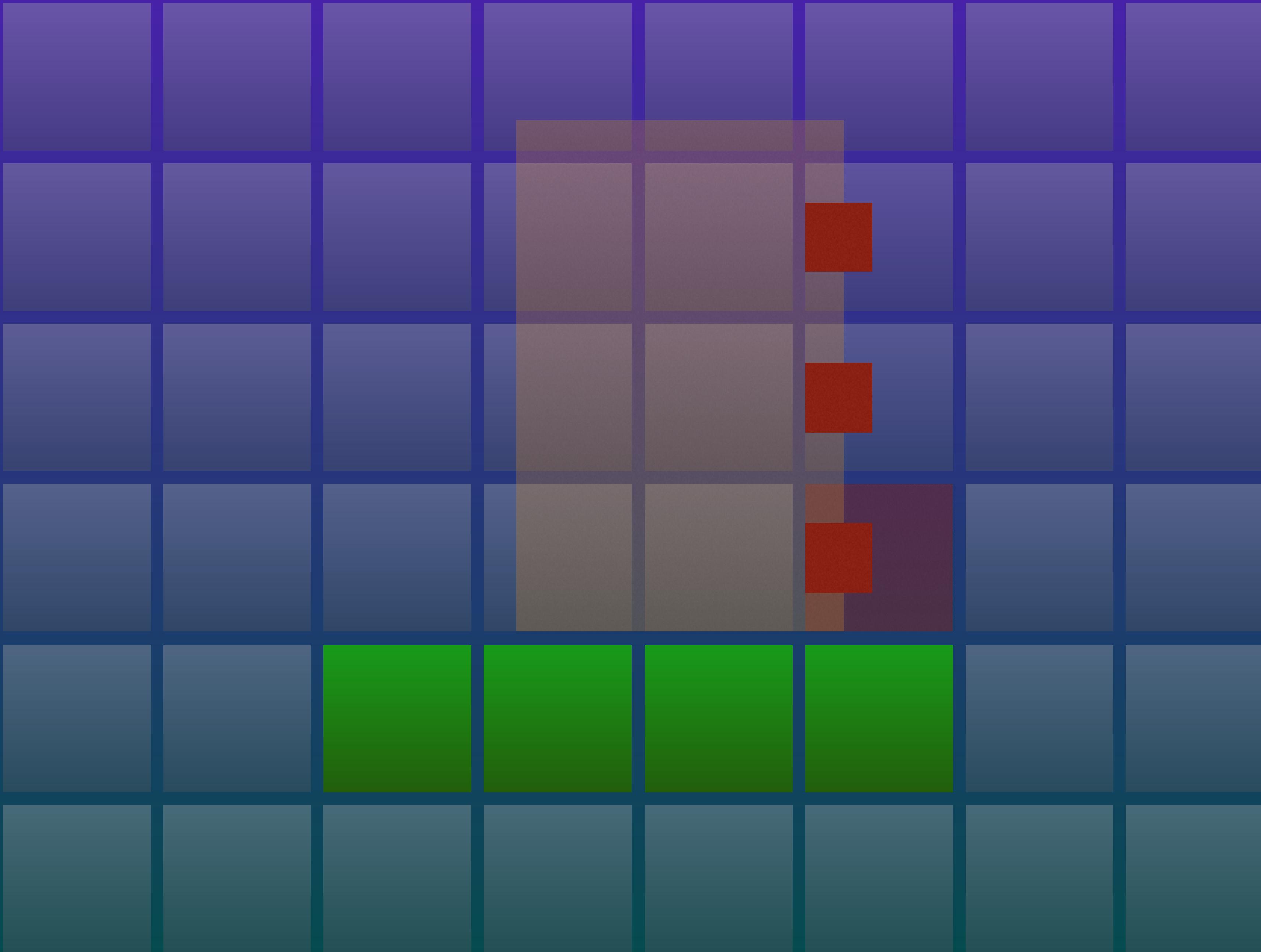


If the tile is solid, adjust our X-coordinate left by the difference between the point we are testing and the left of the tile and set X-velocity to 0, set right collision flag..



You may test additional points along the sprite  
if it is larger than a tile.

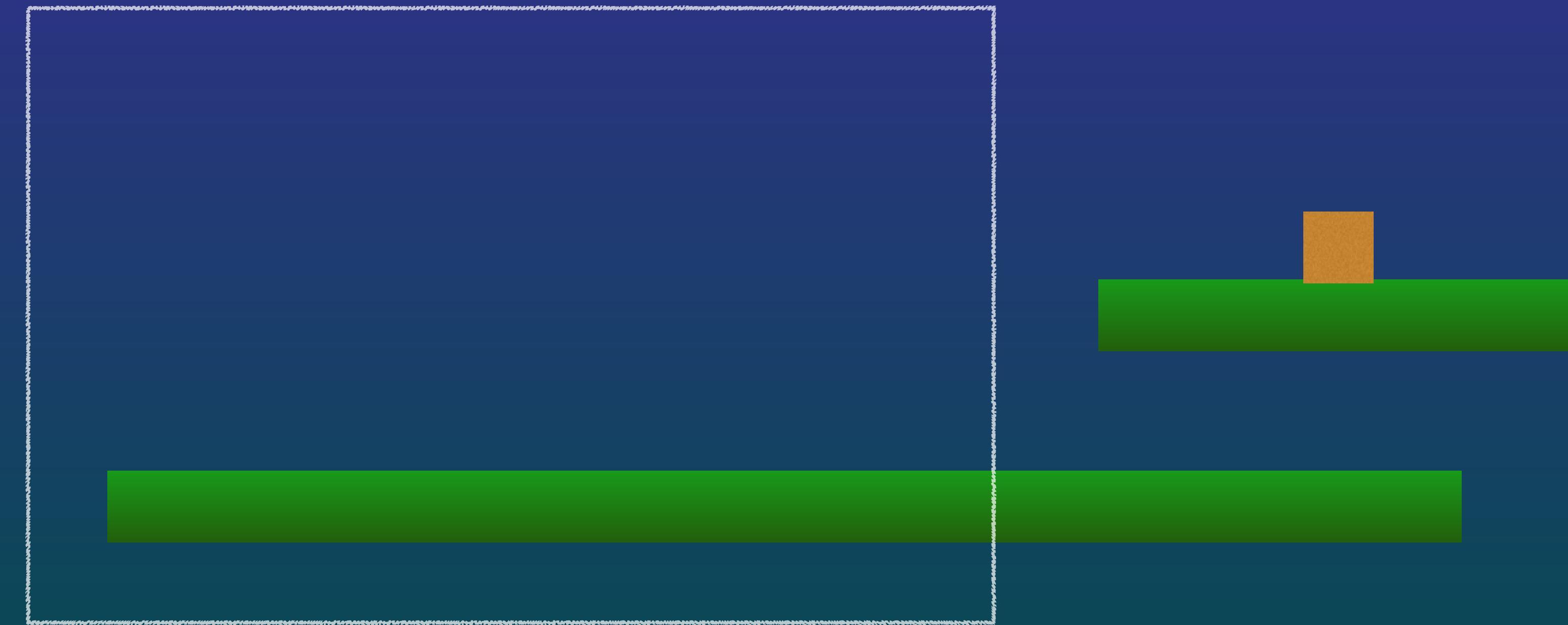




# Scrolling

# What is scrolling?

We need to translate everything so that the player is in the center of the screen.



Call glTranslate before we render anything with the inverse of the player's position.



# The problem with glLoadIdentity()

```
void glPushMatrix();
```

Push the current matrix to the stack.

```
glPushMatrix();
```

```
void glPopMatrix();
```

Pops the matrix from the stack and sets the current matrix to it.

```
glPopMatrix();
```

To transform our entities without affecting the modelview matrix, we must save the current matrix using `glPushMatrix`, do our transforms and render, then restore the matrix using `glPopMatrix`.

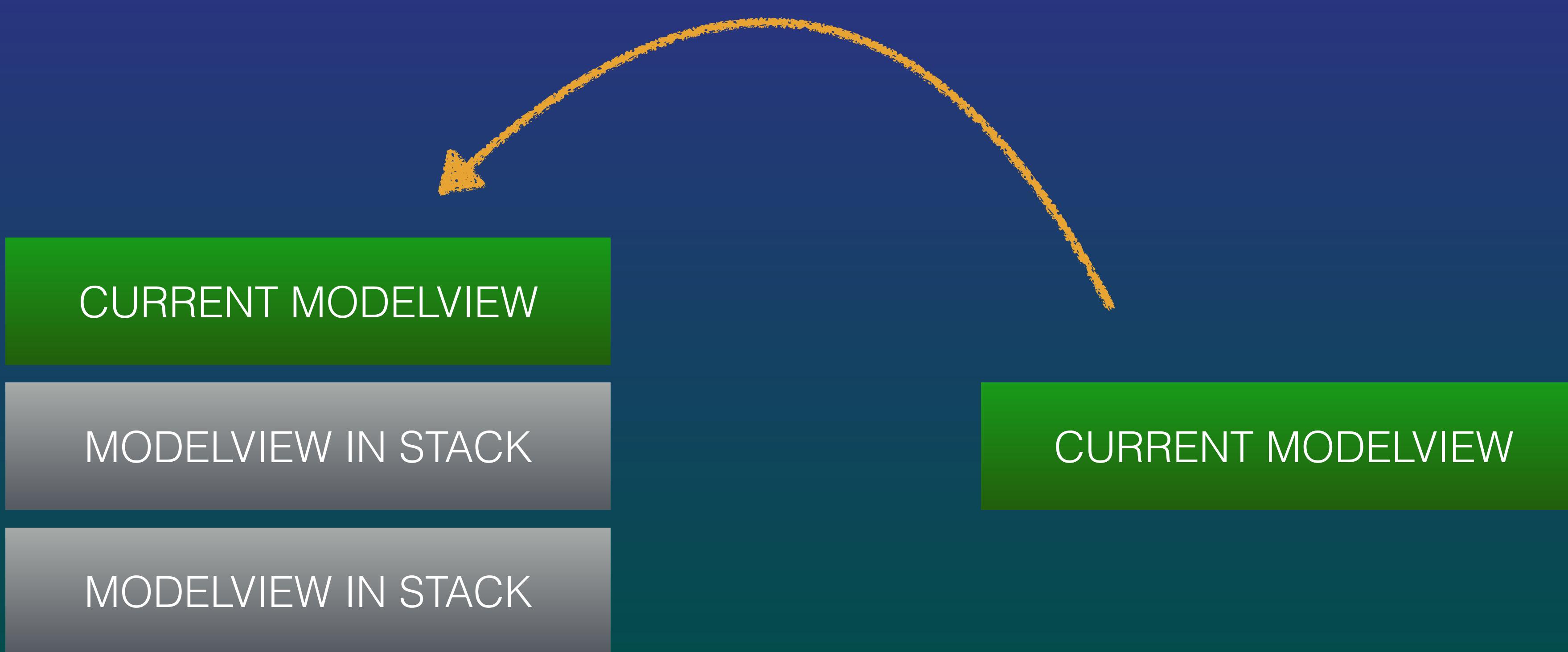
```
glMatrixMode(GL_MODELVIEW);
```

MODELVIEW IN STACK

MODELVIEW IN STACK

CURRENT MODELVIEW

```
glMatrixMode(GL_MODELVIEW);  
glPushMatrix();
```



```
glMatrixMode(GL_MODELVIEW);  
glPushMatrix();  
glTranslatef(x, y, 0.0f);  
// Draw stuff
```

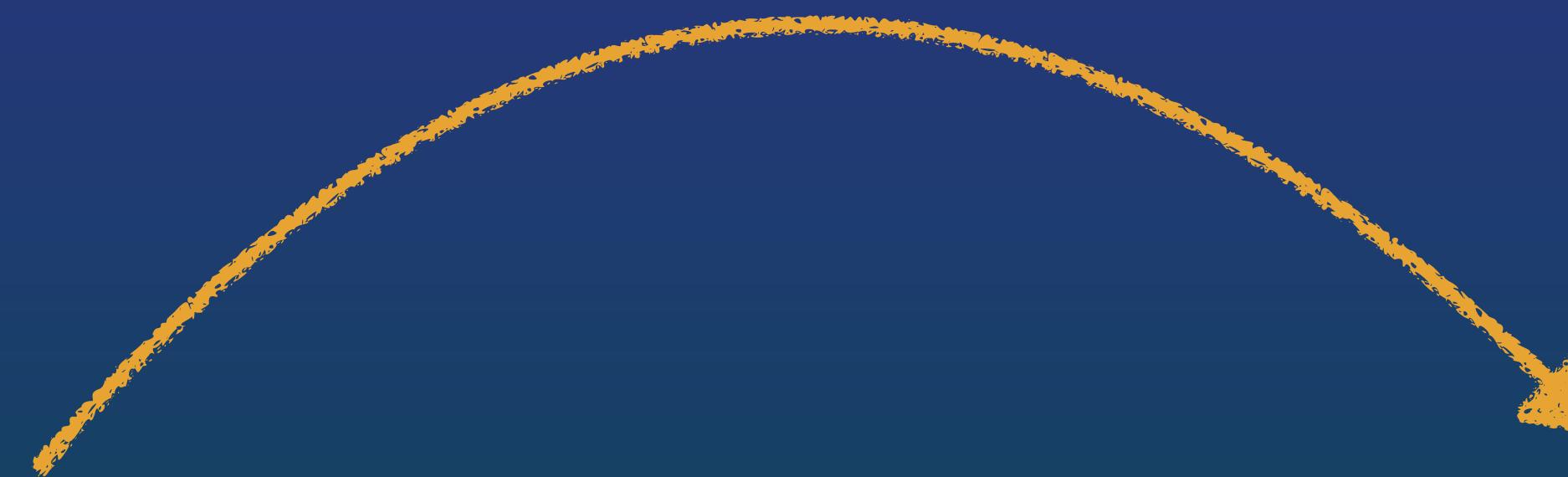
CURRENT MODELVIEW

MODELVIEW IN STACK

MODELVIEW IN STACK

CURRENT MODELVIEW

```
glMatrixMode(GL_MODELVIEW);  
glPushMatrix();  
glTranslatef(x, y, 0.0f);  
// Draw stuff  
glPopMatrix();
```



MODELVIEW IN STACK

CURRENT MODELVIEW

MODELVIEW IN STACK