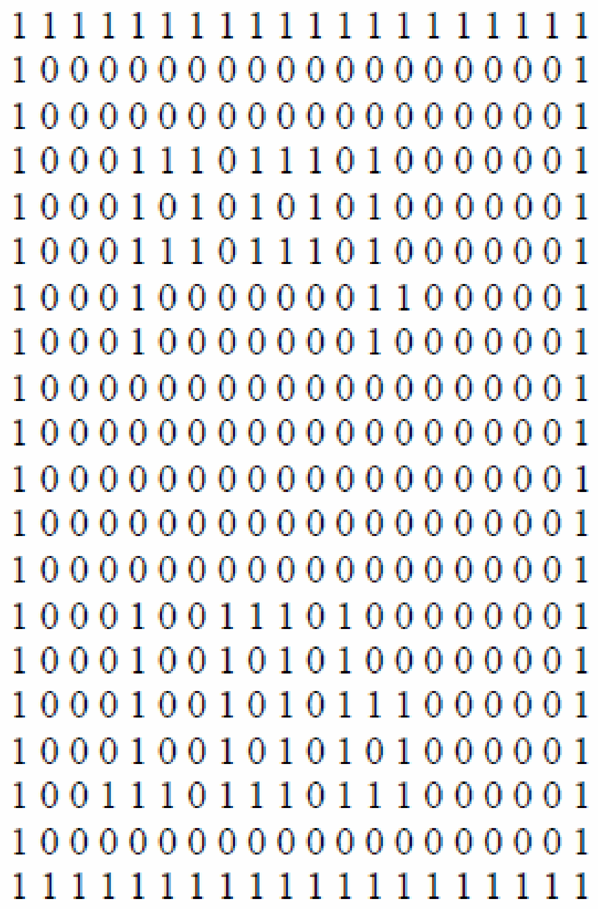
CS230
Game Implementation Techniques

Lecture 11

Outline

- Binary Collision Map
  - Introduction
  - Initialization
- Sprite Collision using Hot Spots
- Snapping
- Normalized Coordinates System

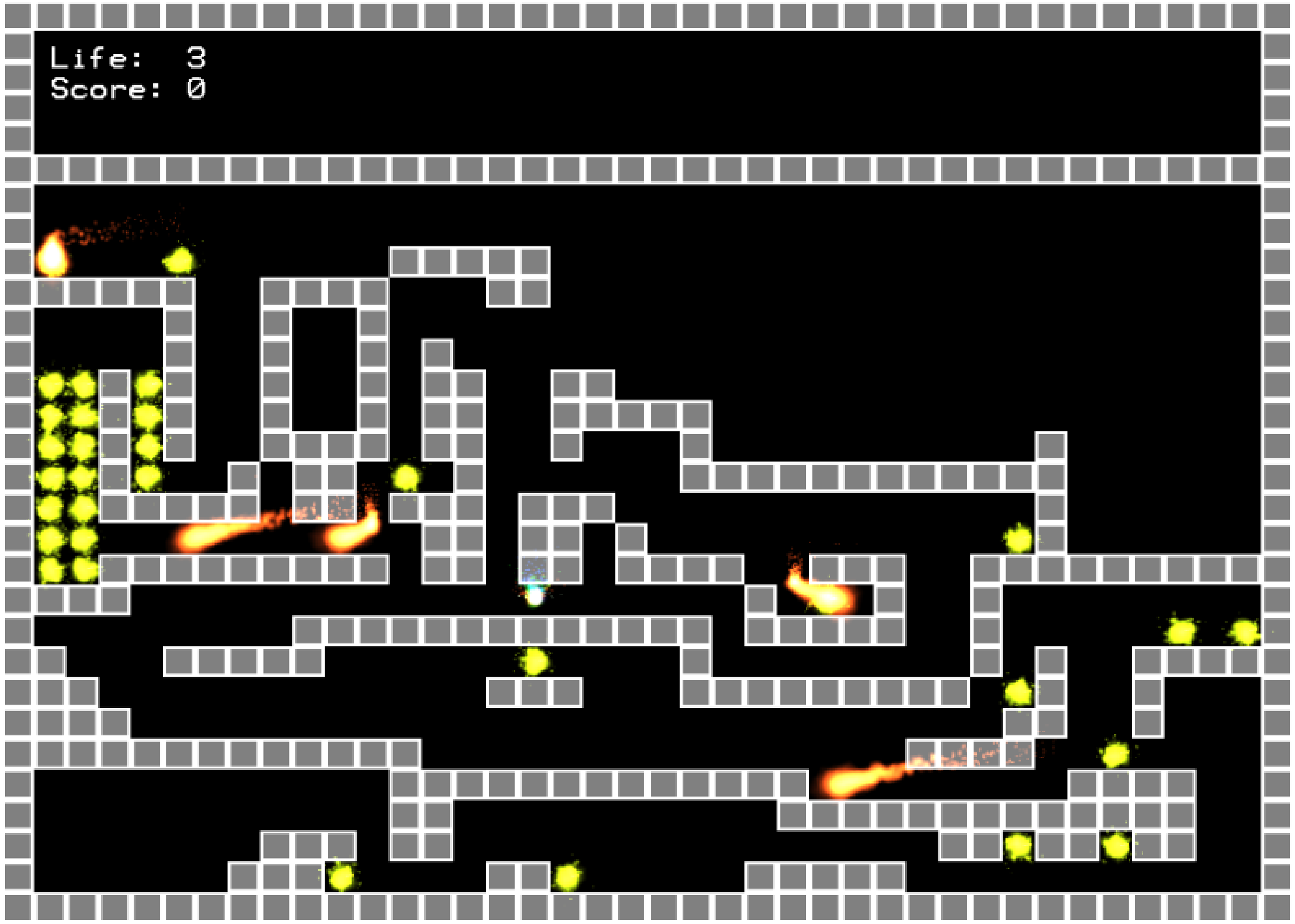
What is Binary Collision Map?



```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 1 1 1 0 1 1 1 0 1 0 0 0 0 0 0 1
1 0 0 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 1
1 0 0 0 1 1 1 0 1 1 1 0 1 0 0 0 0 0 0 1
1 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1
1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 1 0 0 1 1 1 0 1 0 0 0 0 0 0 0 1
1 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 0 0 1
1 0 0 0 1 0 0 1 0 1 0 1 1 1 0 0 0 0 0 1
1 0 0 0 1 0 0 1 0 1 0 1 0 1 0 0 0 0 0 1
1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 0 0 0 0 1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

What Type of Games do we use it for?

Example



Outline

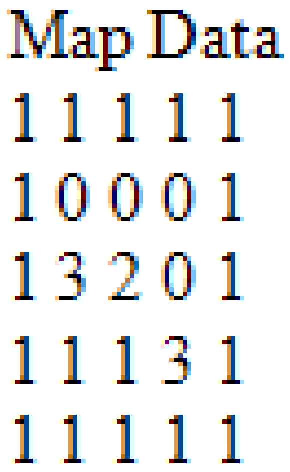
- Binary Collision Map
  - Introduction
  - Initialization
- Sprite Collision using Hot Spots
- Snapping
- Normalized Coordinates System

Binary Collision Map: Initialization (1/2)

- The map should be a grid (which is formed from cells)
- The collision map is a 2D array of “bools”
- Game objects are able to access a cell depending on that cell’s value in the array

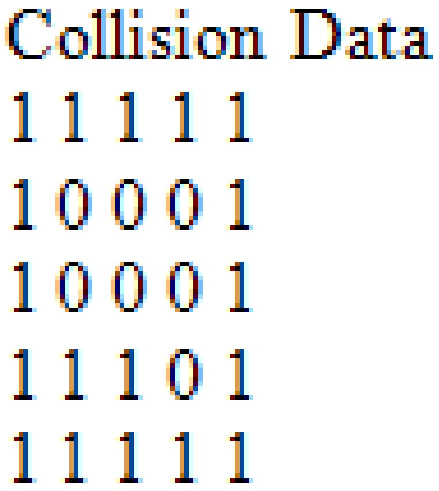
Binary Collision Map: Initialization (2/2)

- Example:



Map Data

1	1	1	1	1
1	0	0	0	1
1	3	2	0	1
1	1	1	3	1
1	1	1	1	1



Collision Data

1	1	1	1	1
1	0	0	0	1
1	0	0	0	1
1	1	1	0	1
1	1	1	1	1

Outline

- Binary Collision Map
  - Introduction
  - Initialization
- Sprite Collision using Hot Spots
- Snapping
- Normalized Coordinates System

Checking for Point Collision (1/2)

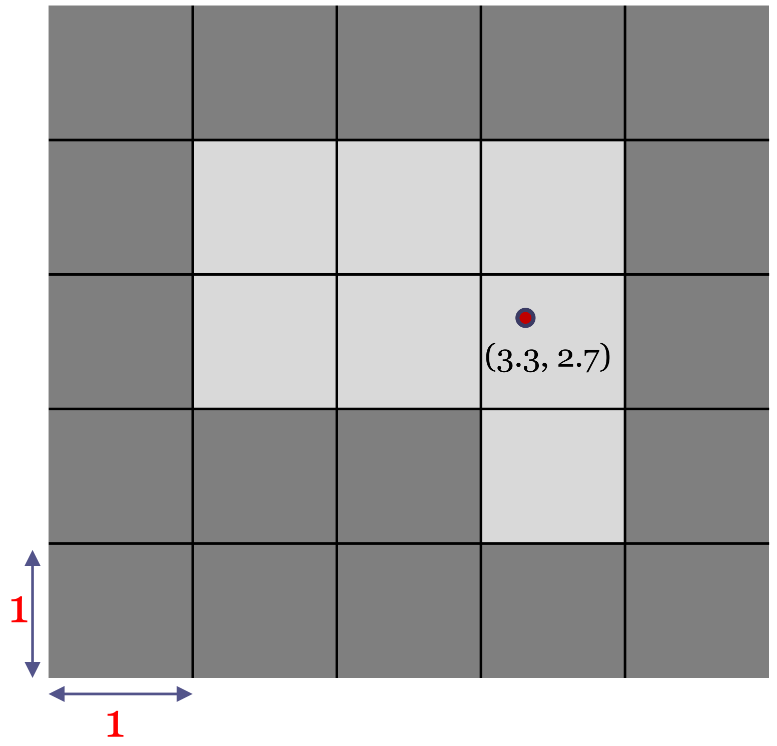
- Knowing that the cell's dimension is (1; 1), to check if a point is in a "solid" cell we get its position in the array (using array indices) and check its value.

Checking for Point Collision (2/2)

- Example:
  - A point is located at (3.3, 2.7)

Collision Data

1 1 1 1 1
1 0 0 0 1
1 0 0 0 1
1 1 1 0 1
1 1 1 1 1

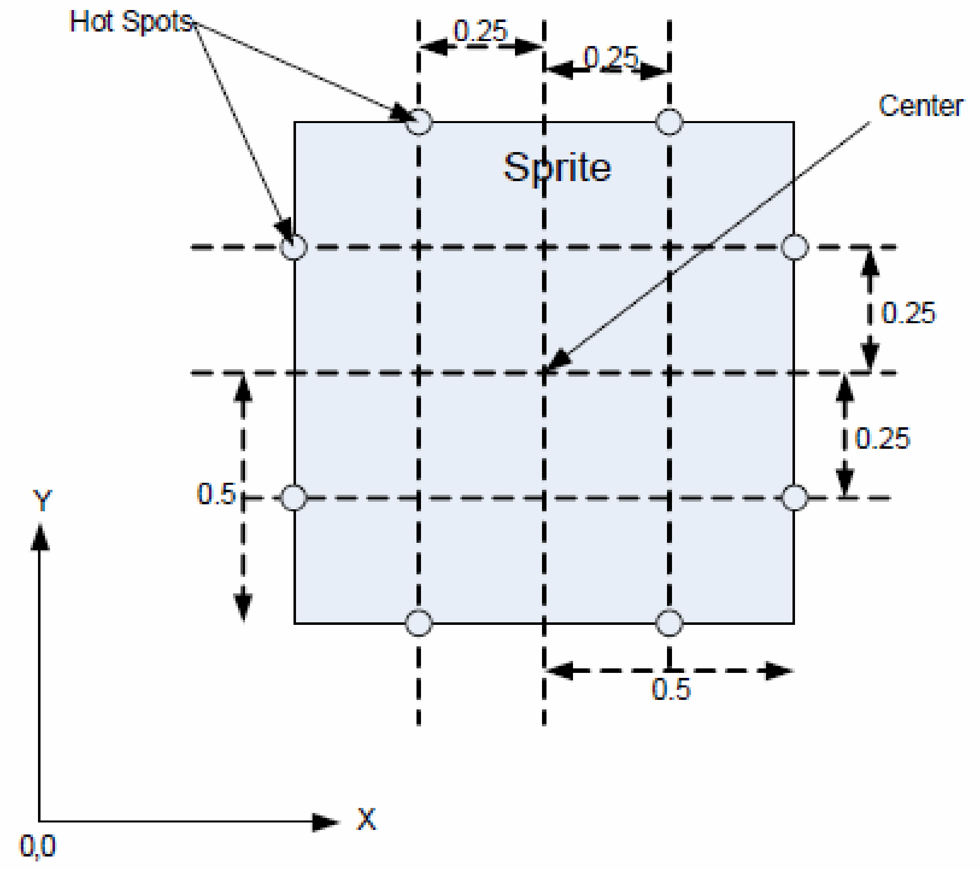


Hot Spots (1/3)

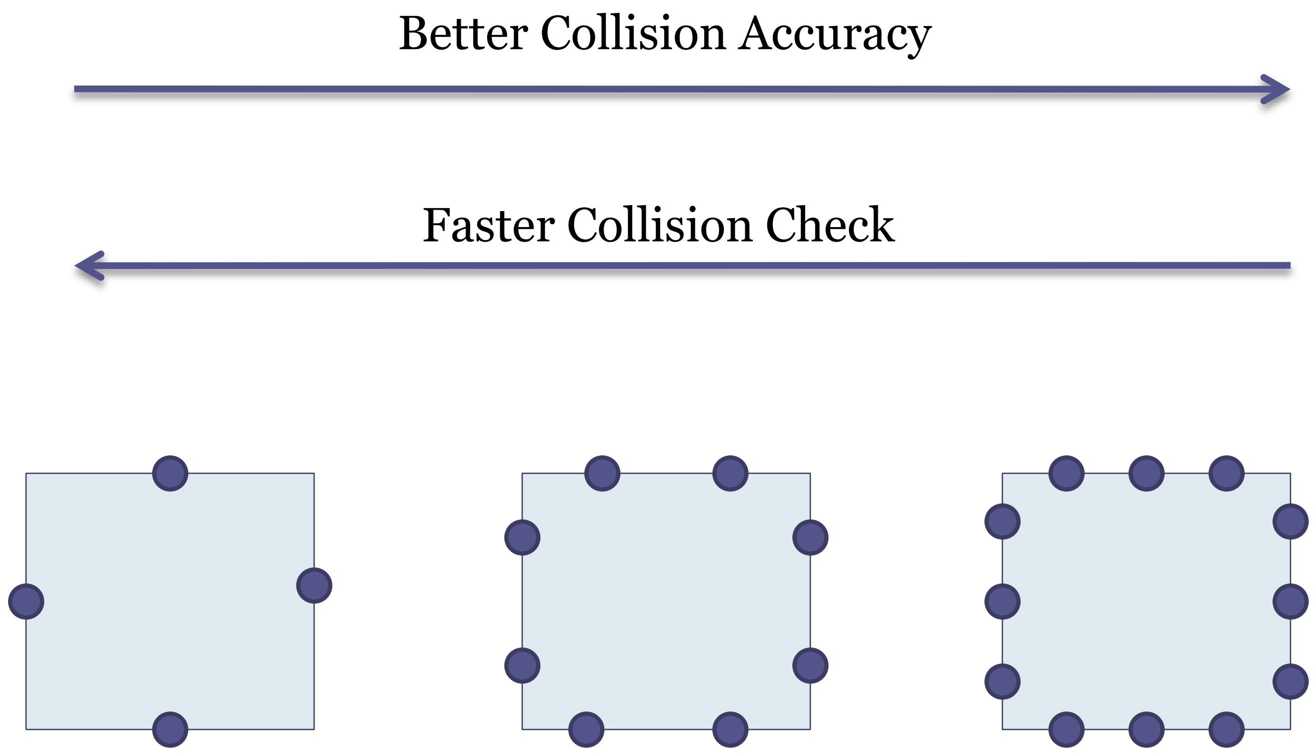
- Our object is not just one point but is encapsulated with a bounding rectangle
- We are dealing with more than one point
- These points are called "Hot Spots"
- Note that this method assumes that both width and height of an object are both 1 (same size of a cell)

Hot Spots (2/3)

- Example:



Hot Spots (3/3)



Sprite Collision using Hot Spots (1/4)

- The collision can occur on the four sides of the sprite (top , bottom, left and right) and on more than one side.

- Each game object instance will have a collision flag, where each bit represents one side.

Sprite Collision using Hot Spots (2/4)

- The least significant bit represents the left side.
- The second bit represents the right side.
- The third bit represents the top side.
- The fourth bit represents the bottom side.

Sprite Collision using Hot Spots (3/4)

- When a certain side is found in a collision state, we set its corresponding bit in the collision flag variable to 1
- How do we set the corresponding bit to 1?

Answer

- This is done by OR-ing the flag with the correspondent collision side value.

- Collision side values

```
#define        COLLISION_LEFT        0x00000001        //0001
#define        COLLISION_RIGHT       0x00000002        //0010
#define        COLLISION_TOP         0x00000004        //0100
#define        COLLISION_BOTTOM      0x00000008        //1000
```

Sprite Collision using Hot Spots (4/4)

- After storing the collision information, how can I check on which side I collided with?
- Answer:
  - This is done by checking the collision flag AND-ed with the correspondent collision side value.

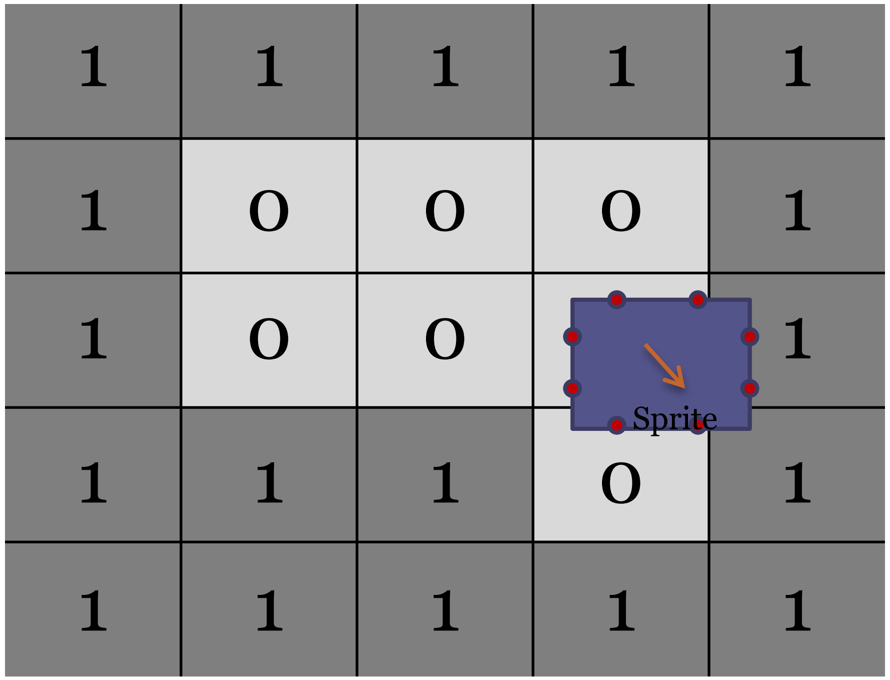
Outline

- Binary Collision Map
  - Introduction
  - Initialization
- Sprite Collision using Hot Spots
- Snapping
- Normalized Coordinates System

Snapping (1/2)

- If at least one hot spot is inside a collision area, we should snap the sprite back to the center of the cell that it belongs to.

Snapping (2/2)



PosX = (int) PosX + 0.5
(Applicable in this example)

PosY = (int) PosY + 0.5
(Not applicable in this example)

Outline

- Binary Collision Map
  - Introduction
  - Initialization
- Sprite Collision using Hot Spots
- Snapping
- Normalized Coordinates System

Normalized Coordinates System (1/3)

- Why?
  - We might want to scale the entire map to the window size, regardless of the grid's width and height.
  - A cell's width might be greater than its height, where the art assets might require that.
  - The cell's width and height directly affect the collision check

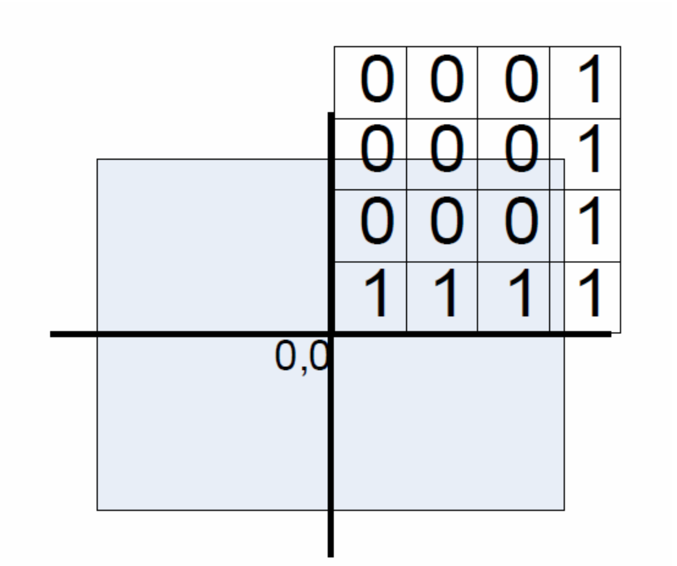
Normalized Coordinates System (2/3)

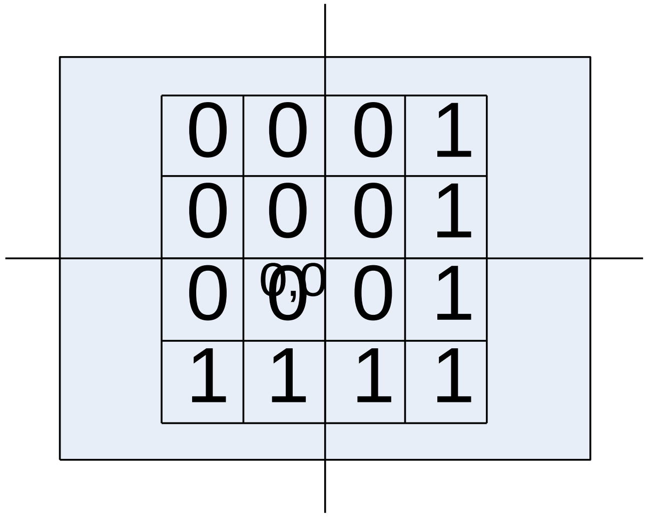
- How?
  - Have the width and height of each cell in the normalized coordinates system to be 1, independently from the final result
  - All the physics (velocity, acceleration, etc…), movement and collision checks are done in the normalized coordinate system

Normalized Coordinates System (3/3)

- Moving our binary map from normalized coordinates system to the world coordinates system requires a transformation matrix made from:
  - Translation
  - Scale

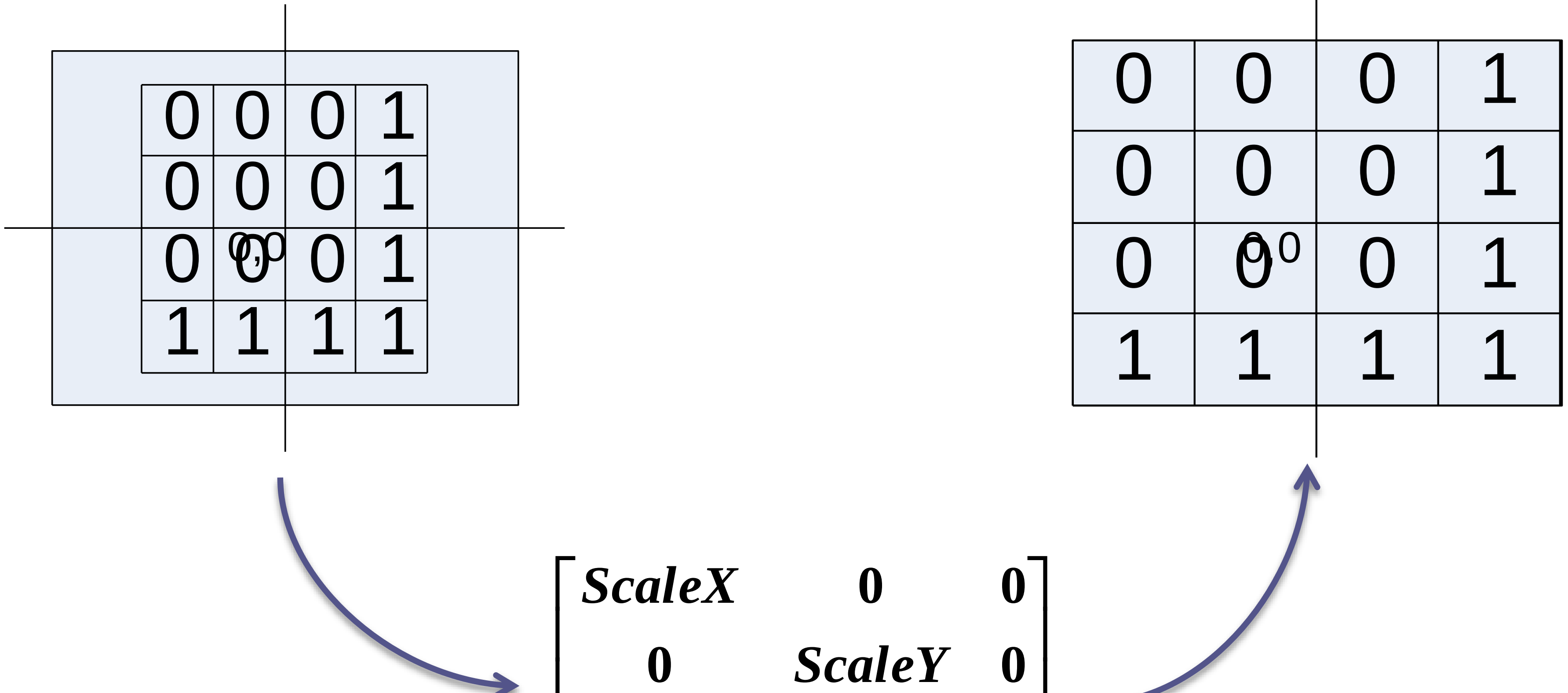
Translation





$$\begin{bmatrix} 1 & 0 & \dfrac{-\text{Grid Width}}{2} \\ 0 & 1 & \dfrac{-\text{Grid Height}}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

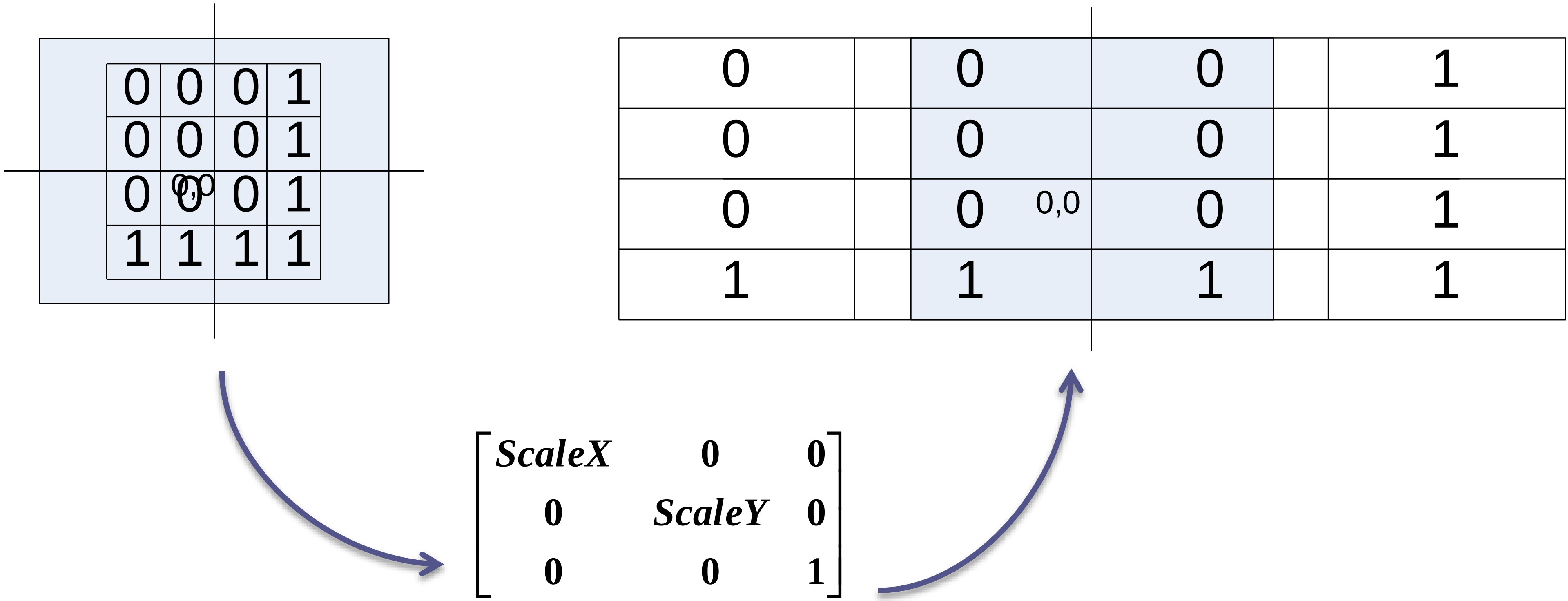
Scale (1/2)



$$\begin{bmatrix} ScaleX & 0 & 0 \\ 0 & ScaleY & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Scale (2/2)

- Scaling could be bigger than the viewport (i.e. scrolling games)



$$\begin{bmatrix} ScaleX & 0 & 0 \\ 0 & ScaleY & 0 \\ 0 & 0 & 1 \end{bmatrix}$$