

Platform Games

Computer Games Development

David Cairns

Introduction

Contents

- This Lecture
 - Tile Maps
- Next Lecture
 - Drawing Sprites
 - Collision Detection
 - Animation Loop

2D Platform Games

The 2D Platform Game

- Player moves through a scrolling world running and jumping between platforms, picking up power-ups and being chased by creatures
- Examples: Sonic the Hedgehog, Super Mario Brothers
- Same principles for a variety of games
- Puzzles, monsters & power ups make the difference

Core Elements

- The player: a sprite
- The game world: a tile map + scrolling background image
- Power ups: animated tiles in the game map or sprites
- Monsters: more sprites
- Collision handler

Drawing the Game World

We have a number of choices for the game world:

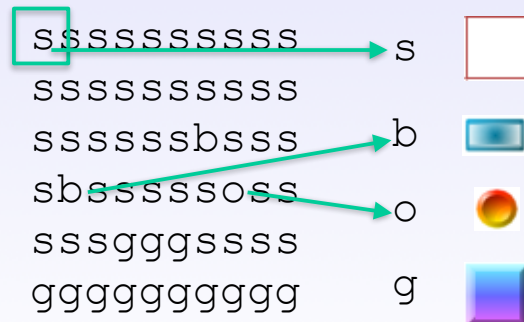
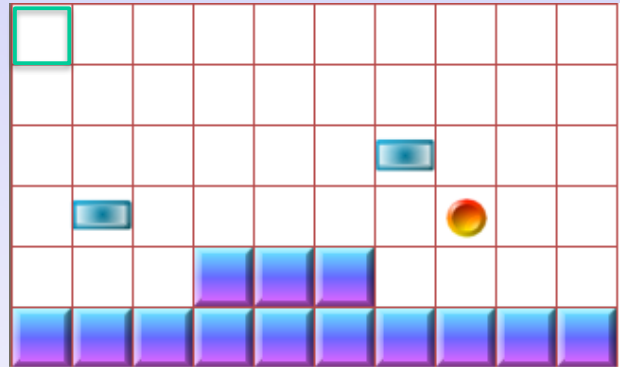
- Draw a complete image of the entire 2D game world and show the player the bit they are on
 - High memory requirement
 - Requires specific code to detect if the player has collided with relevant parts of the image
- Compose the game world from a small number of tiles
 - Low memory requirement
 - Very flexible
 - Provides useful context information for collision detection

We will look at Tile Maps

Tile Maps

Tile Maps

- A tile map consists of a coded grid where each symbol in the grid is represented by a particular tile in the game
- The complete map is loaded into a 2D array where each cell references the relevant image
- The section of the tile map the player is on is then displayed
- Very compact representation
- Very flexible
 - It is very easy to alter map cells during the game
- Each game level uses a different map



Tile Maps

Code Requirements

- Load a Tile Map
- Display Tile Map
- Alter Tile Map

Loading A Tile Map

Example Tile Map Format

```
10 5 32 32
// The first line should contain the width and height of the
// map and the width and height of each tile. A list of character to
// tile mappings is then provided where each character is preceded by a
// # character. The dot character always defaults to a blank space
// Note that the referenced files should be in the same directory as the
// tile map.
#b=orangeblock.png
#c=greencircle.png
#g=glasses.png
// The actual tile map is preceded by the #map line
#map
bbbbbbbbbbb
b.....b
b..g.....b
bcccccccb
bbbbbbbbbbb
```

Loading a Tile Map

Java Code

```
public boolean loadMap(String folder, String mapfile)
{
    ... Read in the map and tile dimensions
    ... Read in the character to tile mappings and store them

    // Now read in the tile map structure
    if (trimmed.startsWith("#map"))
    {
        int row=0;
        while ((line = in.readLine()) != null)
        {
            if (line.trim().startsWith("//")) continue;

            for (int col=0; col<mapWidth && col<line.length(); col++)
                tmap[col][row] = new Tile(line.charAt(col),col*tileWidth,row*tileHeight);
            row++;
        }
    }
    ...
}
```

Displaying a Tile Map

Java Code (from handout)

```
public void draw(Graphics2D g, int xoff, int yoff)
{
    if (g == null) return;

    Image img=null;
    Rectangle rect = (Rectangle)g.getClip();
    int xc,yc;

    for (int r=0; r<mapHeight; r++)
    {
        for (int c=0; c<mapWidth; c++)
        {
            img = getTileImage(c, r);
            if (img == null) continue;
            xc = xoff + c*tileWidth;
            yc = yoff + r*tileHeight;

            // Only draw the tile if it is on screen, otherwise go back round the loop
            if (xc+tileWidth < 0 || xc >= rect.x + rect.width) continue;
            if (yc+tileHeight < 0 || yc >= rect.y + rect.height) continue;
            g.drawImage(img,xc,yc,null);
        }
    }
}
```

Altering a Tile Map

Java Code (from handout)

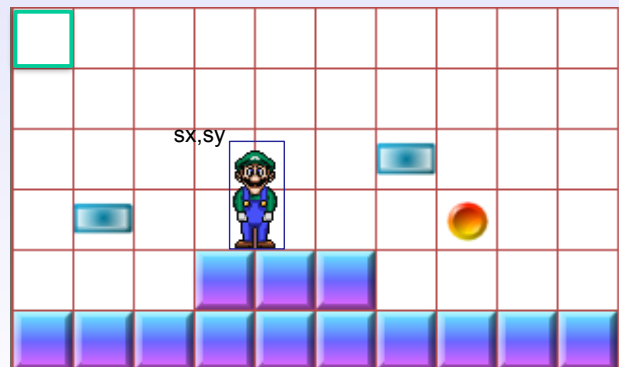
```
public char getTileChar(int x, int y)
{
    if (!valid(x,y)) return '?';
    return tmap[x][y].getCharacter();
}

public boolean setTileChar(char ch, int x, int y)
{
    if (!valid(x,y)) return false;
    tmap[x][y].setCharacter(ch);
    return true;
}
```

Sprites & Tile Maps

A common task will be to work out which tiles a sprite is colliding with. It will usually be sitting on more than one tile but we consider the simple case first by looking at the top left corner of a sprite.

- First we find the sprite's top left coordinates:
 - `sx = s.getX(), sy = s.getY()`
- Then we find out how wide and how tall a tile is
 - `tileWidth = tmap.getTileWidth(), tileHeight = tmap.getTileHeight()`
- If we divide the sprite's x coordinate by the width of a tile, we will get the number of tiles across the x axis that the sprite is positioned at
 - `xtile = (int)(sx / tileWidth)`
- The same applies to the y coordinate
 - `ytile = (int)(sy / tileHeight)`
- Example:
 - `sx = 35, sy = 22`
 - `tileWidth = 10, tileHeight = 10`
 - `xtile = 35/10 = 3`
 - `ytile = 22/10 = 2`
 - If you count from 0...
- Note: This is for the top left corner, you will need to look at each corner of your sprite.



Tile Collision Code

```
/**
 * Check and handles collisions with a tile map for the given sprite 's'. Initial functionality is limited...
 *
 * @param s          The Sprite to check collisions for
 * @param tmap       The tile map to check
 */
public void tileCollision(Sprite s, TileMap tmap)
{
    // Take a note of a sprite's current position
    float sx = s.getX();
    float sy = s.getY();

    // Find out how wide and how tall a tile is
    float tileWidth = tmap.getTileWidth();
    float tileHeight = tmap.getTileHeight();

    // Divide the sprite's x coordinate by the width of a tile, to get
    // the number of tiles across the x axis that the sprite is positioned at
    int xtile = (int)(sx / tileWidth);
    // The same applies to the y coordinate
    int ytile = (int)(sy / tileHeight);

    // What character is at this position?
    char ch = tmap.getTileChar(xtile, ytile);

    if (ch != '.') // If it's not a dot (empty space), handle it
    {
        // Here we just stop the sprite.
        s.stop();
        // You should move the sprite to a position that is not colliding
    }

    // We need to consider the other corners of the sprite
    ...
}
```