

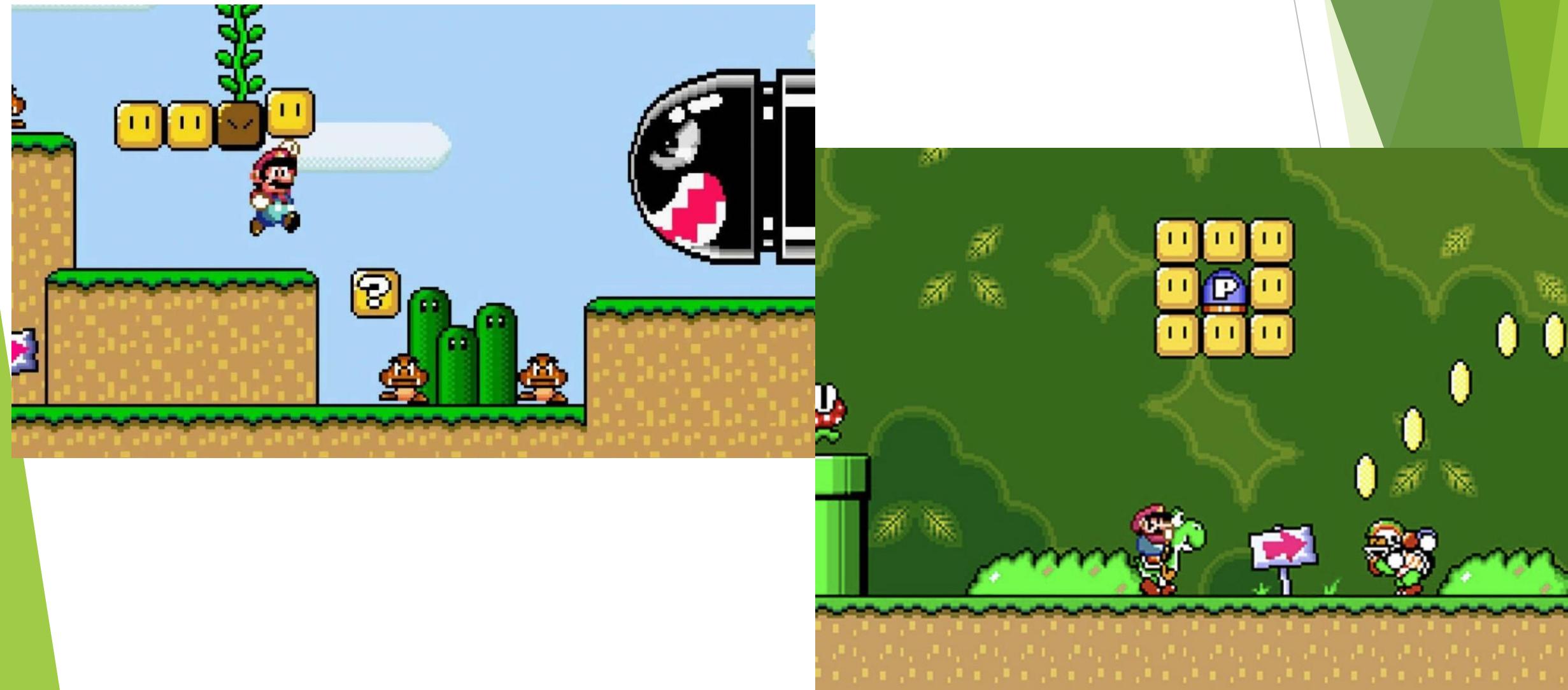
**SUPER
MARIO**

TM



Super Mario Bros

What makes Super Mario unique?



Key characteristics (requirements)

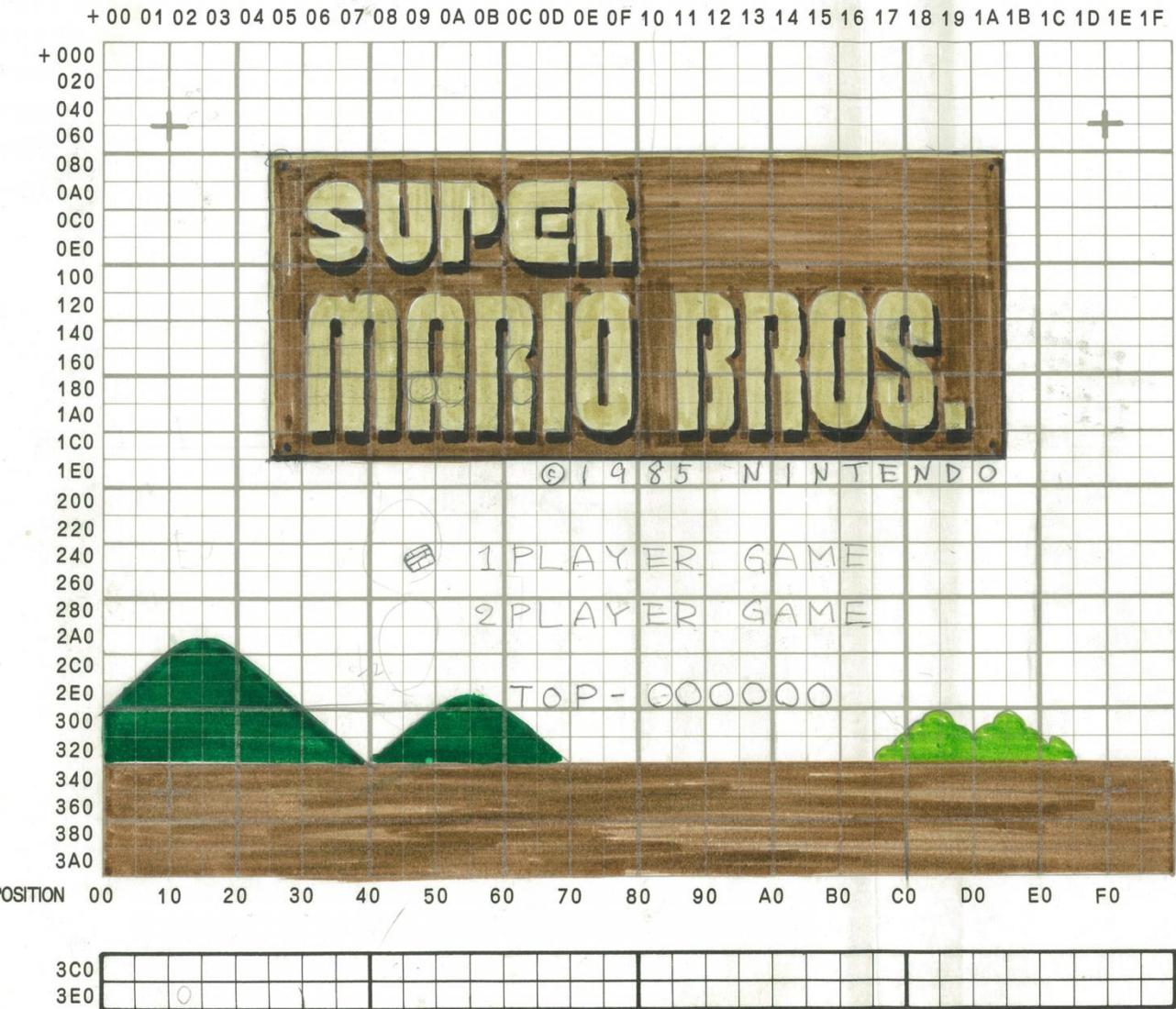
- ▶ Side scroller: Mario/Camera moves right
- ▶ Ground tiles + levels to climb up
- ▶ Gravity effect of jump/double jump
- ▶ Collects coins and optional powerups
- ▶ Enemies (Goombas) try to take lives.
- ▶ Time limit in which Mario must reach the end of the level



Paper
design!

BG1	2000H	1	0	3	2	T
BG2	2400H					
BG3	2800H					
BG4	2C00H	5	4	7	6	

BG PLANNING SHEET	PHASE	年月日	テザイナ		プロクラマー
MEMO: 地上基本 BG の D を使、2E11		AM	時	分	



0 POSITION

COLOR GENERATOR DATA TABLE

B.G. COLOR

FRAME COLOR	CL D1:D0	CHAR m:m	C.G. Addr	C.G. DATA	VIDEO COLOR	REMARKS
NONE	0 0	0 0	3FOOH			
		0 1	01			
		1 0	02			
		1 1	03			
BROWN	0 1	0 0	04			
		0 1	05			
		1 0	06			
		1 1	07			
RED	1 0	0 0	08			
		0 1	09			
		1 0	0A			
		1 1	0B			
ORANGE	1 1	0 0	0C			
		0 1	0D			
		1 0	0E			
		1 1	0F			

OBJ COLOR

	CL D1:D0	CHAR m:m	C.G. Addr	C.G. DATA	VIDEO COLOR	REMARKS
	0 0	0 0	3F10H			
		0 1	11			
		1 0	12			
		1 1	13			
	0 1	0 0	14			
		0 1	15			
		1 0	16			
		1 1	17			
	1 0	0 0	18			
		0 1	19			
		1 0	1A			
		1 1	1B			
	1 1	0 0	1C			
		0 1	1D			
		1 0	1E			
		1 1	1F			

Paper design

- BG1 2000H
 - BG2 2400H
 - BG3 2800H
 - BG4 2000H
 1 0 3 2 1
 5 4 1 7 6

BG PLANNING SHEET

MEMO: W1 - 2

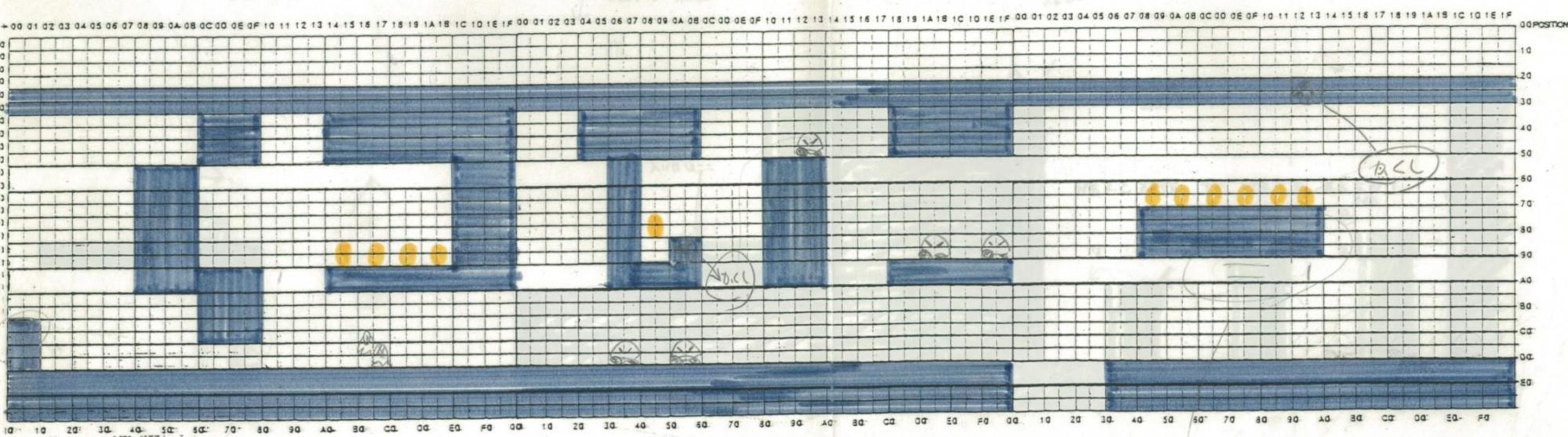
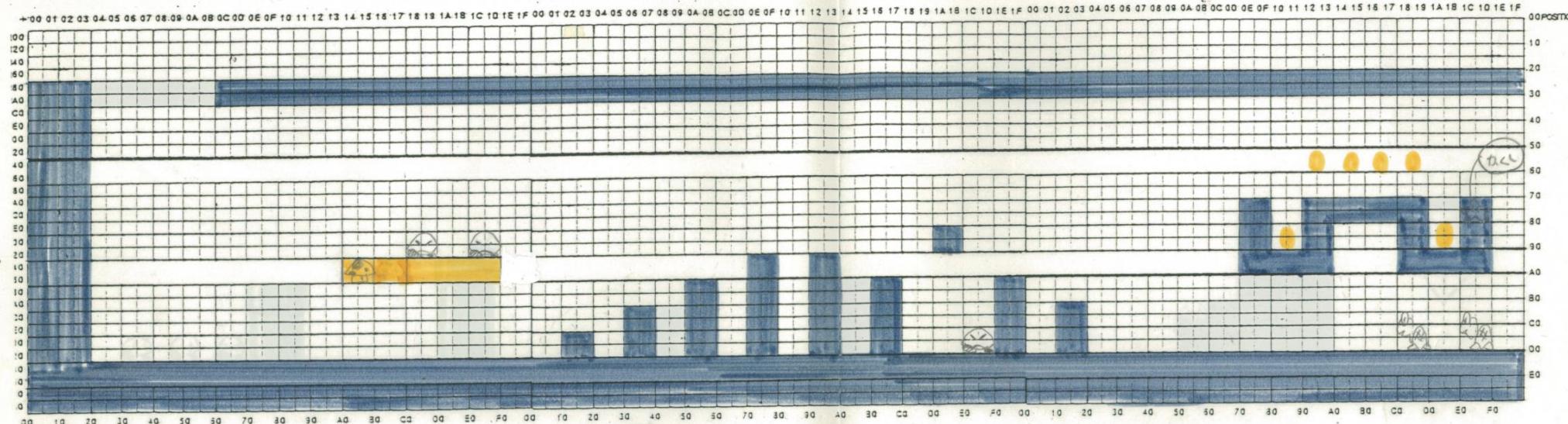
PHASE

年月日
テ
サ
イ
ナ

AM 時 分
P
M
I

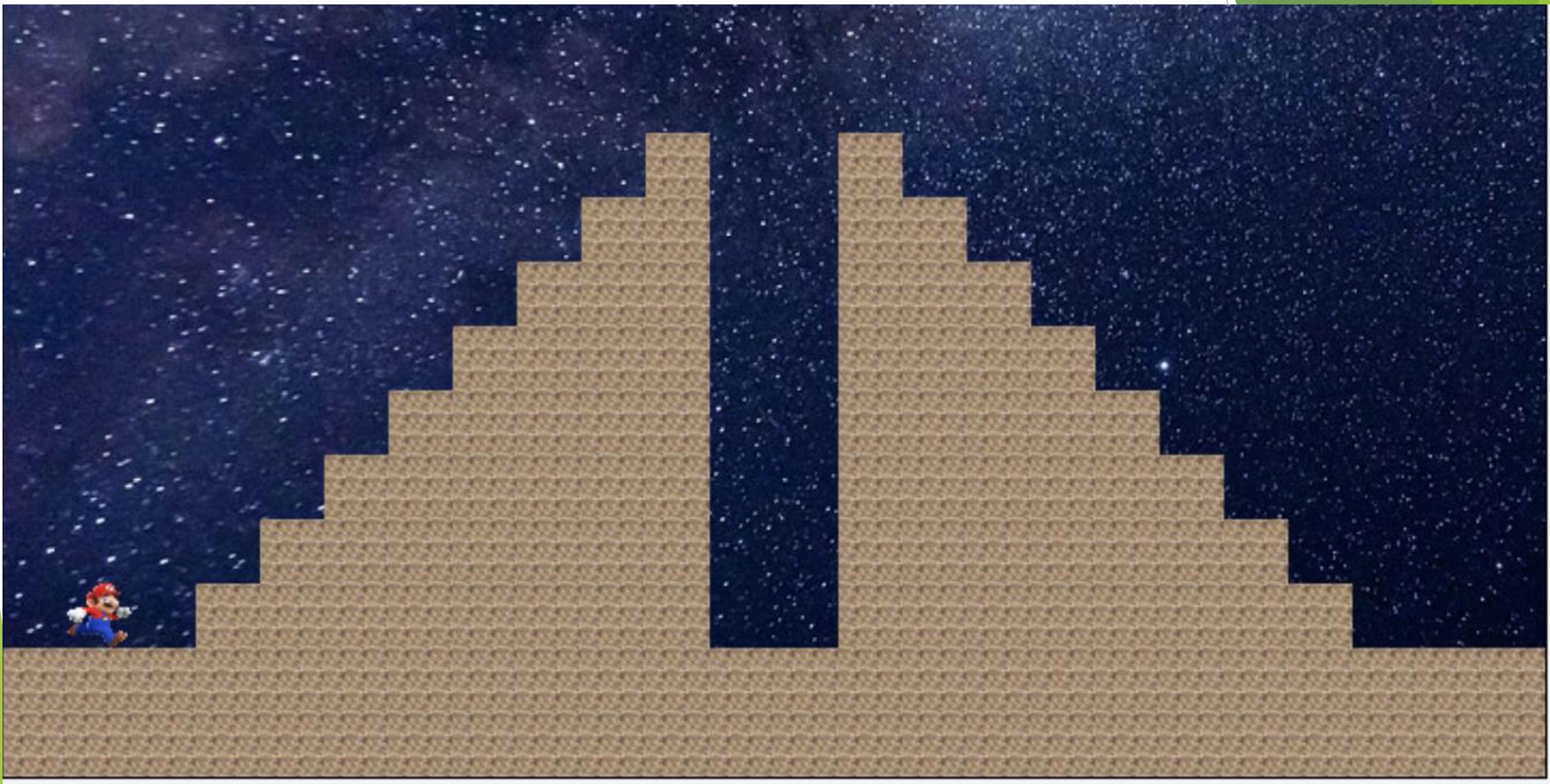
ブ
ロ
ク
ラ
シ
ア

< Under ground - Q >



How do we design a solution to this?

- ▶ Mario would move right (optionally left if levels allows)
 - ▶ Move coordinates (x) when respond to left and right key press
- ▶ The array of blocks
 - ▶ 2D array of graphics?
- ▶ Moving camera with Mario
 - ▶ An x coordinate to keep a track of this?
 - ▶ Compare x/y coordinates?
- ▶ Jump/double jump - gravity
 - ▶ Change the x and y coordinates accordingly





0

1

2

3

4

5

6

7

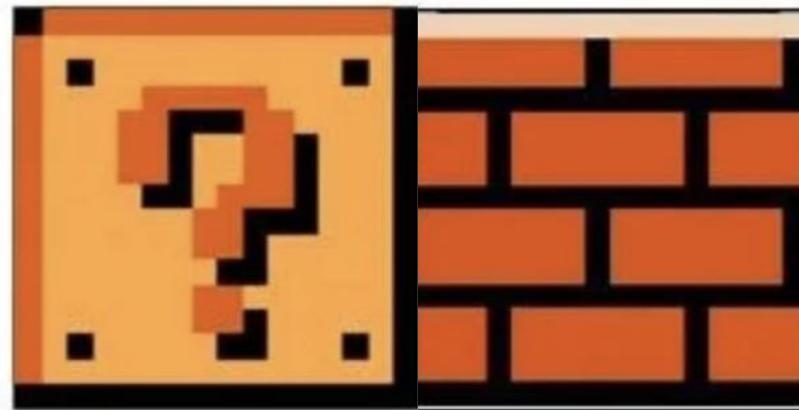
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

0	15	16	17											
1														
2														
3														
4														
5														
6														
7														

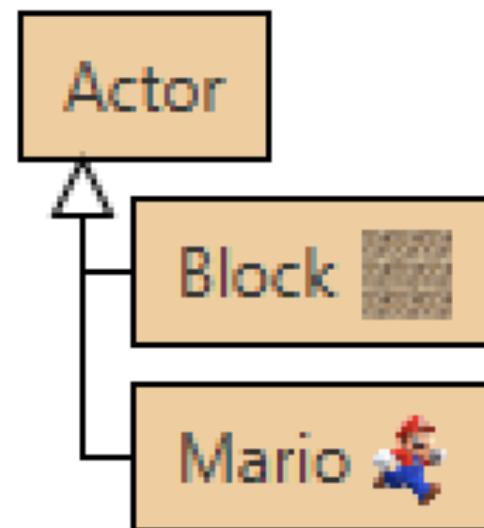


What about OOP?

- ▶ Should we build some of these entities as classes?



OOP in Greenfoot



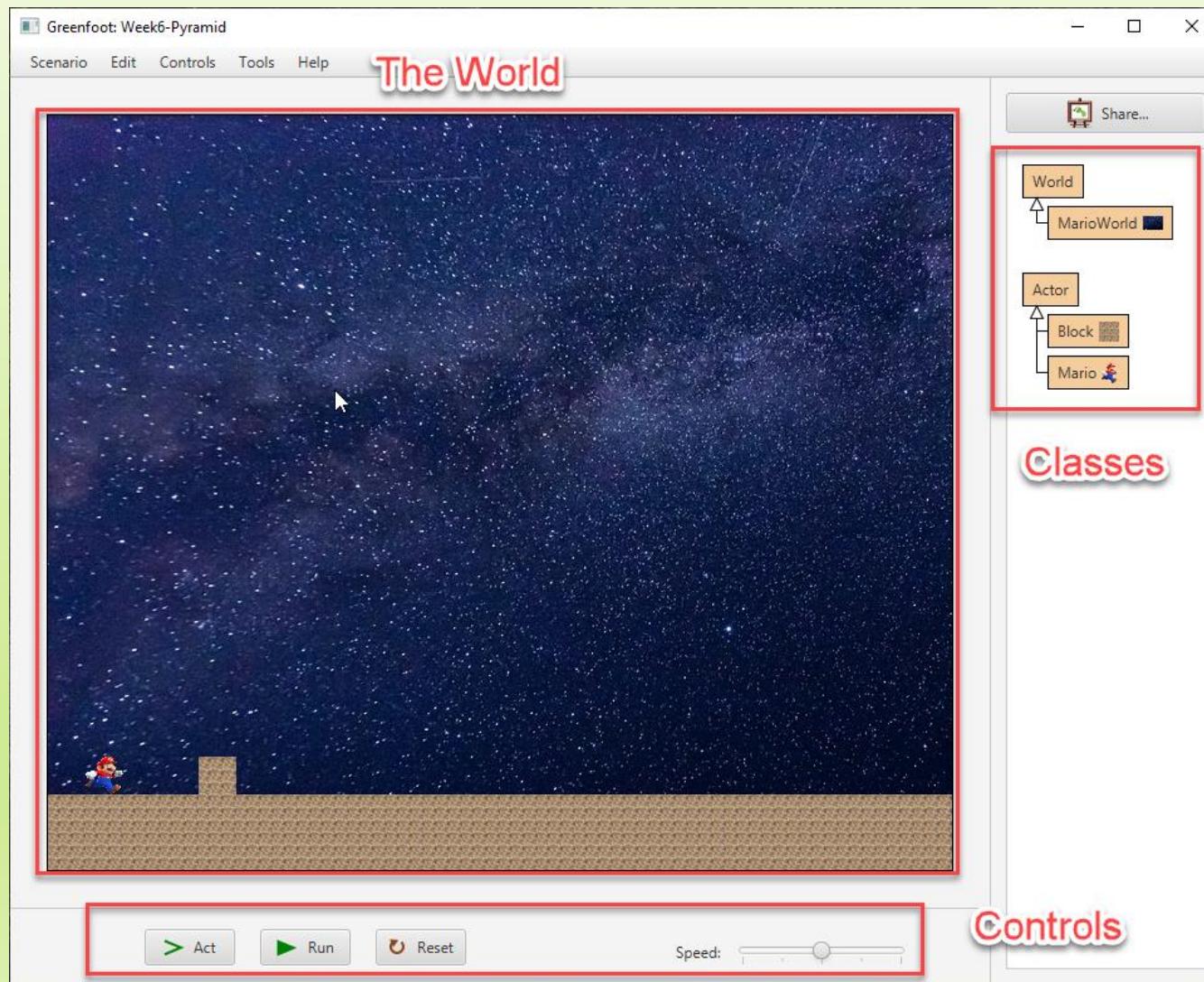
Implementation



Mario Pyramid (Worlds) in Greenfoot

By Derek Peacock

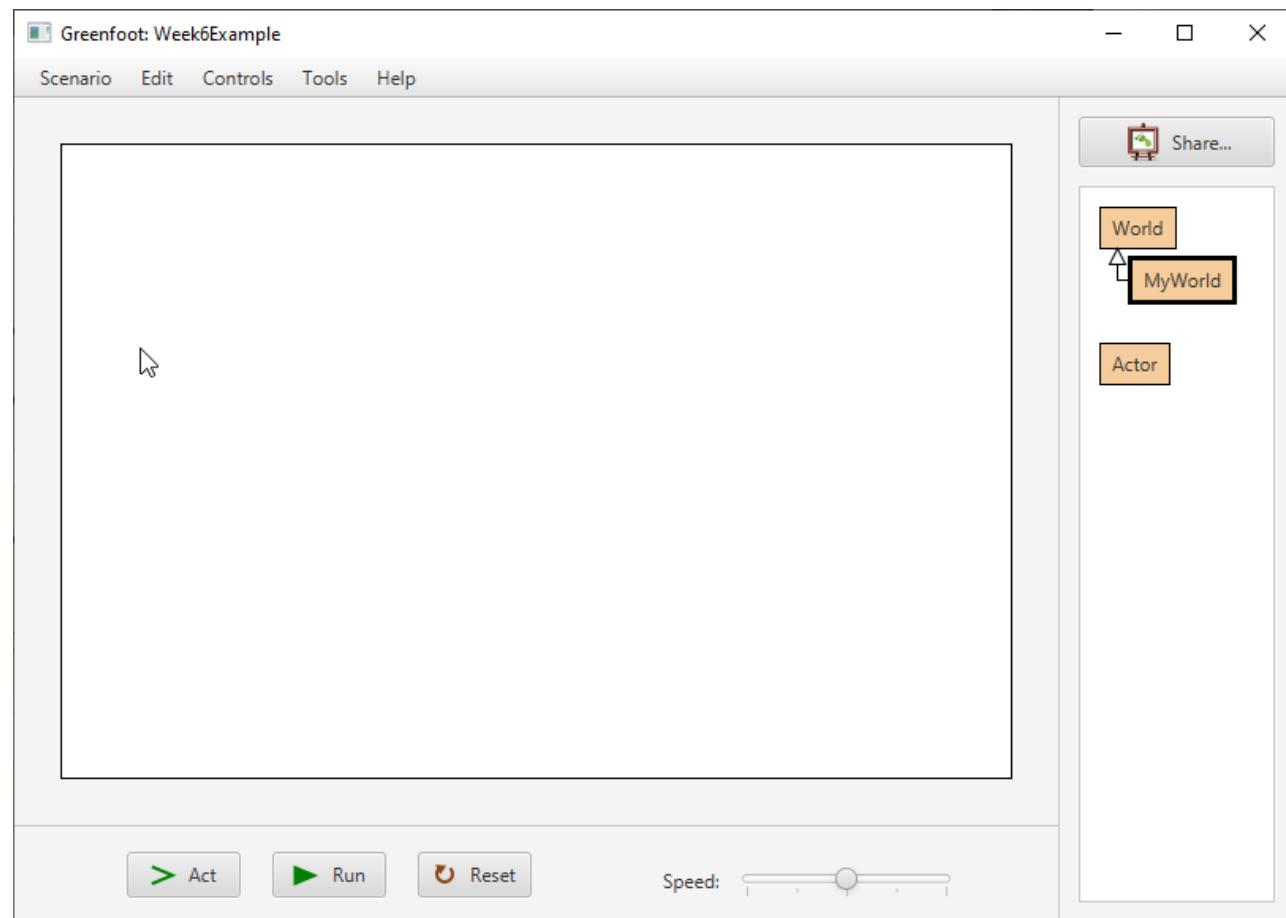
The Greenfoot System



MarioWorld is a kind of World

Mario is a kind of Actor
Block is a kind of Actor

Starting a new Greenfoot Project

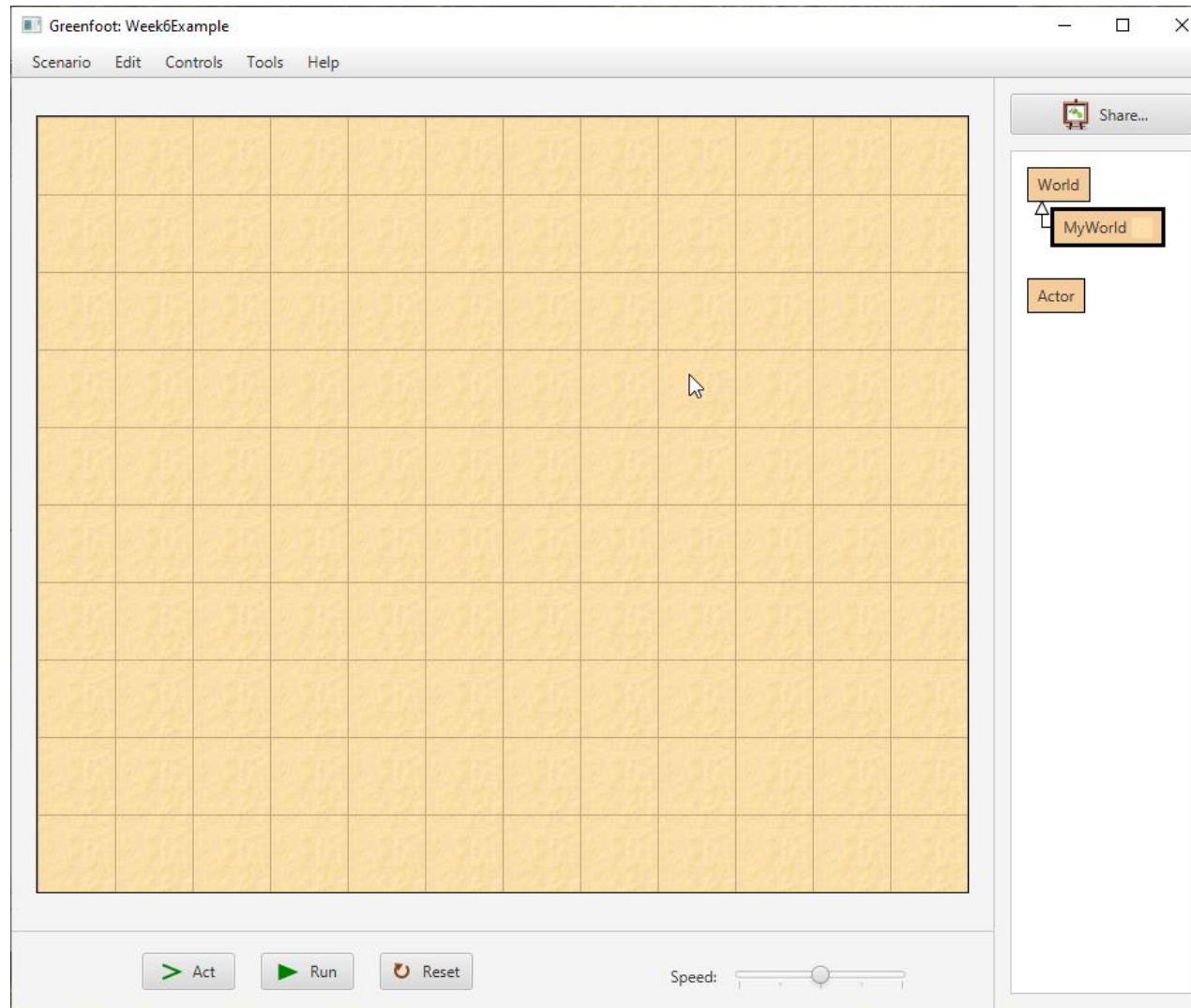


The code editor window title is "MyWorld - Week6Example". The menu bar includes "Class", "Edit", "Tools", and "Options". The toolbar includes "Compile", "Undo", "Cut", "Copy", "Paste", "Find...", "Close", and "Source Code". The code editor displays the following Java code:

```
1 import greenfoot.*; // (World, Actor, GreenfootImage, Greenfoot)
2
3 /**
4  * Write a description of class MyWorld here.
5  *
6  * @author (your name)
7  * @version (a version number or a date)
8 */
9 public class MyWorld extends World
10 {
11
12     /**
13      * Create a new world with 600 x 400 cells
14      * with a cell size of 1 x 1 pixels.
15      */
16     public MyWorld()
17     {
18         super(600, 400, 1);
19     }
20 }
```

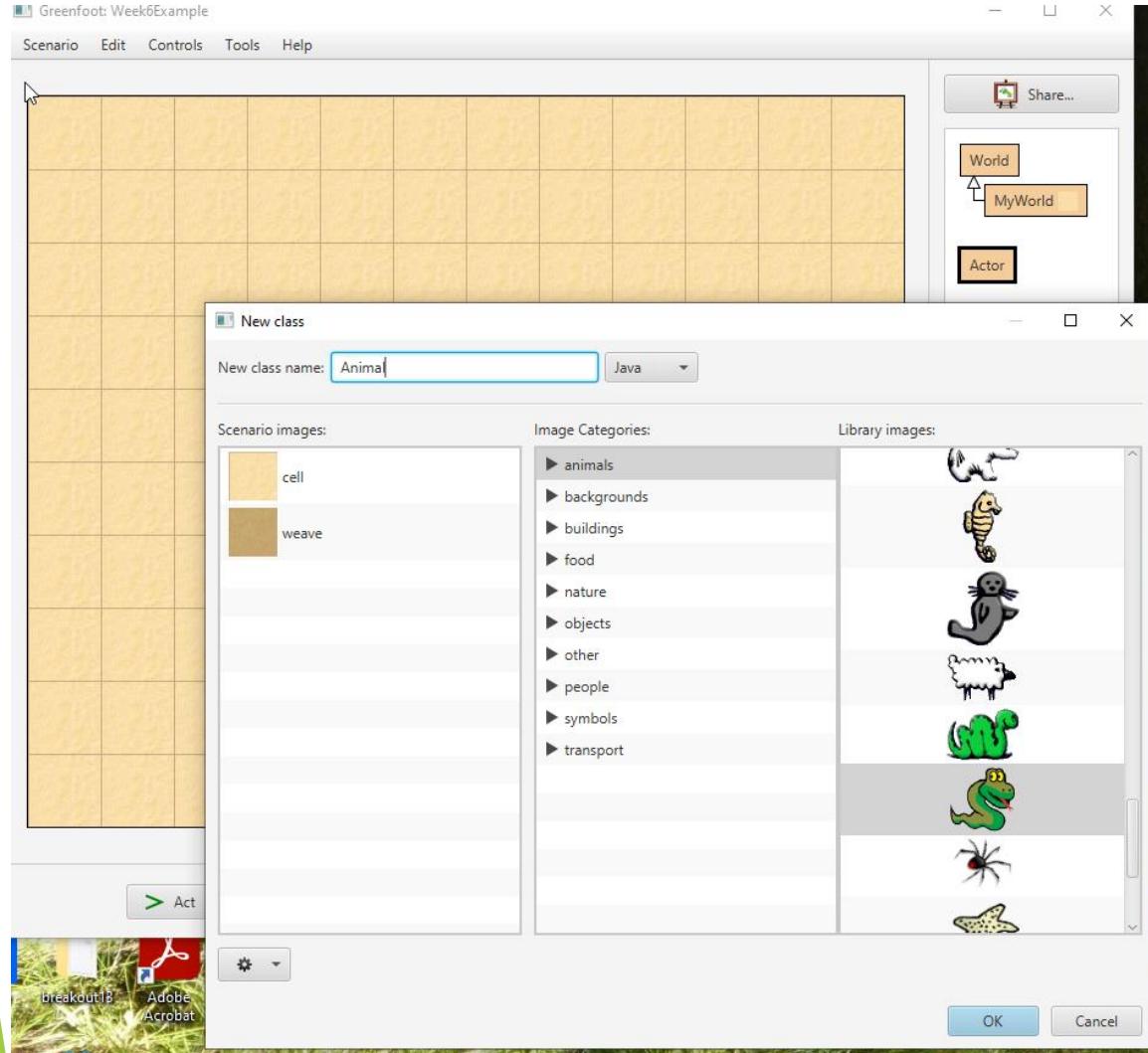
At the bottom of the code editor, a status bar message reads "Class compiled - no syntax errors".

Setting up the World



- ▶ Right click on MyWorld and select an appropriate image.
- ▶ The cells in this image do not match the cells in the world!!
- ▶ It is just a background image!

Adding an Actor



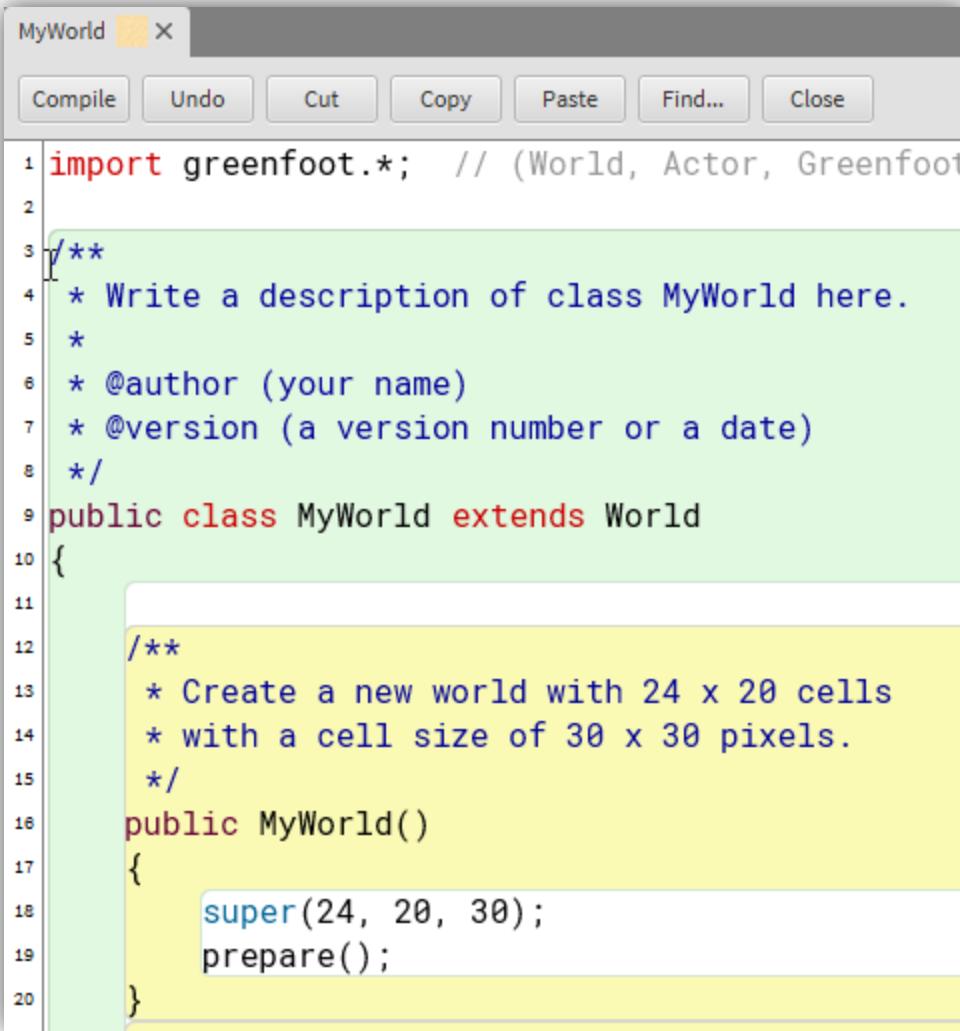
- ▶ Right click on **Actor** and Add a Subclass
- ▶ Give it a class name
- ▶ Select an image

Creating an instance of the Actor

The screenshot shows the Greenfoot IDE interface. On the left is the scenario window titled "Greenfoot: Week6Example" showing a 24x20 grid world with three green snakes. The snakes are positioned at approximately (4, 8), (10, 12), and (15, 4). On the right is the code editor window titled "MyWorld - Week6Example" with the class "MyWorld" selected. The code creates three instances of the "Animal" class and adds them to the world:

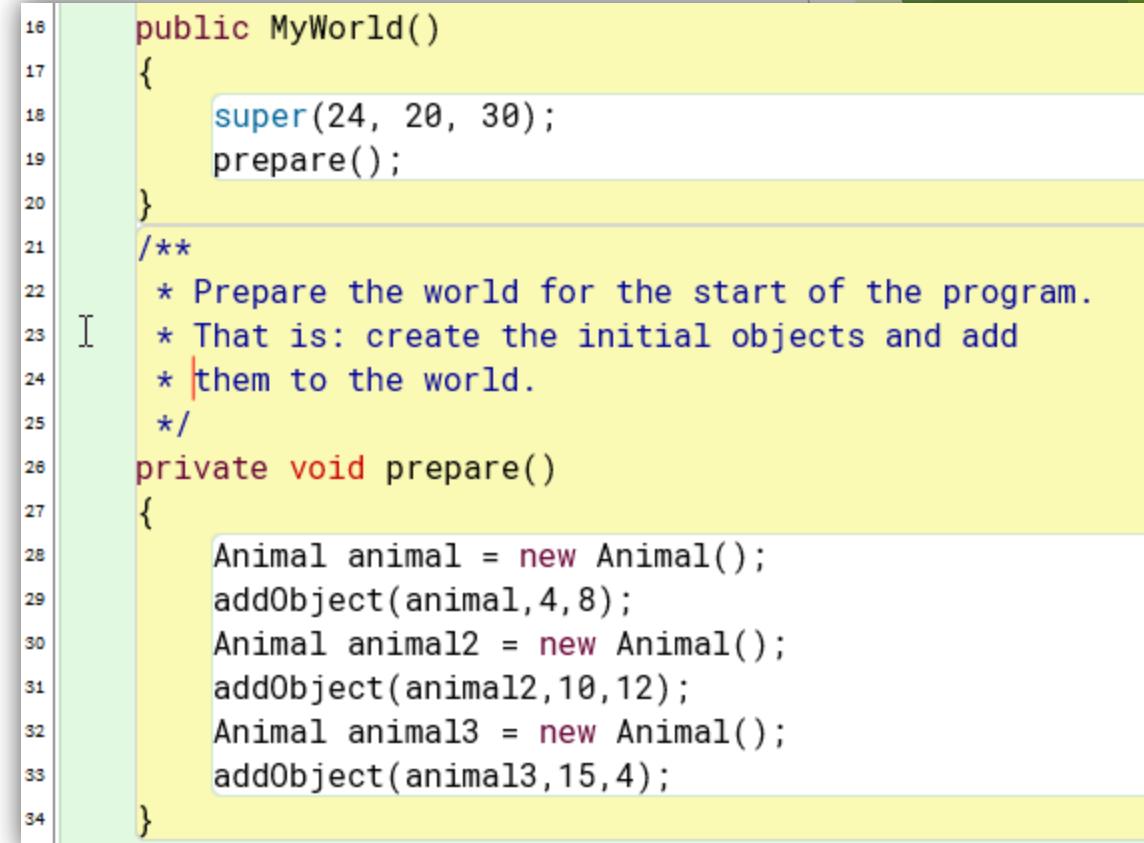
```
/**  
 * Create a new world with 24 x 20 cells  
 * with a cell size of 30 x 30 pixels.  
 */  
public MyWorld()  
{  
    super(24, 20, 30);  
    prepare();  
}  
/**  
 * Prepare the world for the start of the program.  
 * That is: create the initial objects and add  
 * them to the world.  
 */  
private void prepare()  
{  
    Animal animal = new Animal();  
    addObject(animal, 4, 8);  
    Animal animal2 = new Animal();  
    addObject(animal2, 10, 12);  
    Animal animal3 = new Animal();  
    addObject(animal3, 15, 4);  
}
```

The World has been saved!!!



A screenshot of the Greenfoot IDE showing the code for the `MyWorld` class. The window title is `MyWorld`. The code includes imports, a class comment, and a constructor that calls `super(24, 20, 30)` and `prepare()`.

```
1 import greenfoot.*; // (World, Actor, Greenfoot
2
3 /**
4  * Write a description of class MyWorld here.
5  *
6  * @author (your name)
7  * @version (a version number or a date)
8  */
9 public class MyWorld extends World
{
10
11     /**
12      * Create a new world with 24 x 20 cells
13      * with a cell size of 30 x 30 pixels.
14      */
15     public MyWorld()
16     {
17         super(24, 20, 30);
18         prepare();
19     }
20 }
```

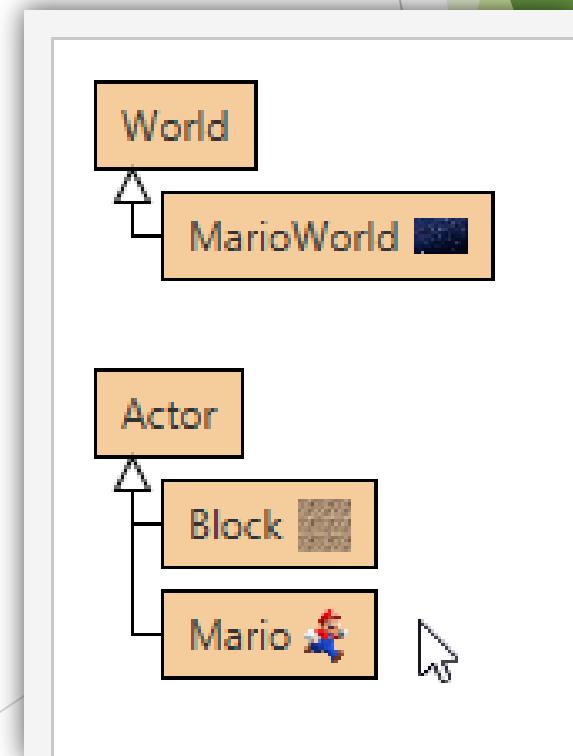
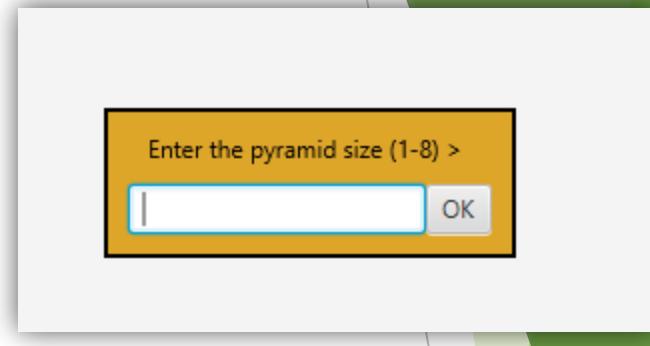
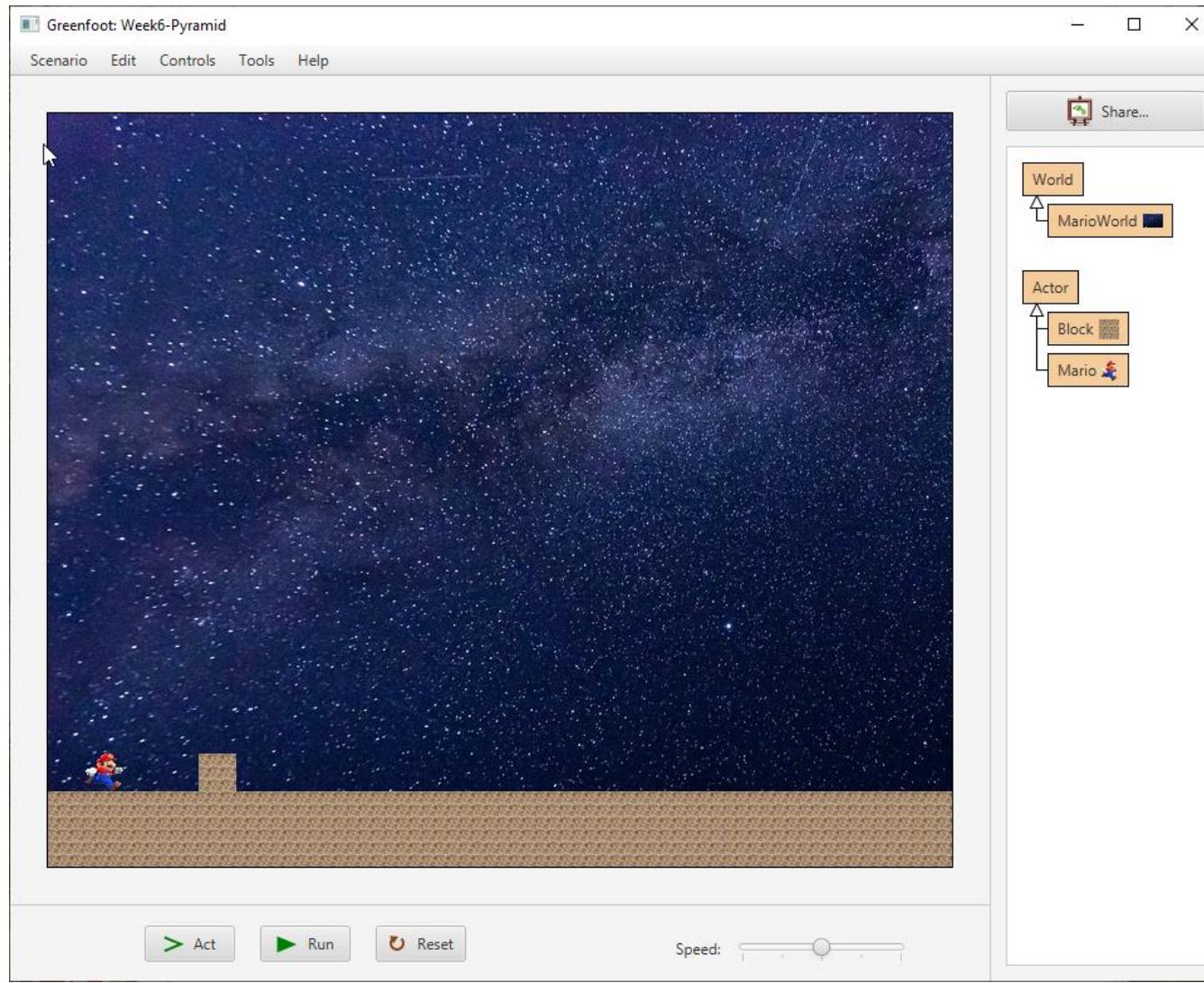


A diagram illustrating a class hierarchy. The `MyWorld` class has a constructor that calls `super(24, 20, 30)` and a `prepare()` method. The `prepare()` method creates three `Animal` objects and adds them to the world at specific coordinates.

```
16 public MyWorld()
17 {
18     super(24, 20, 30);
19     prepare();
20 }
21 /**
22  * Prepare the world for the start of the program.
23  * That is: create the initial objects and add
24  * them to the world.
25  */
26 private void prepare()
27 {
28     Animal animal = new Animal();
29     addObject(animal, 4, 8);
30     Animal animal2 = new Animal();
31     addObject(animal2, 10, 12);
32     Animal animal3 = new Animal();
33     addObject(animal3, 15, 4);
34 }
```

I would name the method
`setup()` or `setupAnimals()`

Your Starter for 10



Starting Code

```
public class MarioWorld extends World
{
    public static final int MAXN_COLUMNS = 24;
    public static final int MAXN_ROWS = 20;
    public static final int GROUND_ROW = 17;
    public static final int TILE_SIZE = 30; // pixels

    private Mario mario;

    public MarioWorld()
    {
        // Create a new world with 24 x 20 tiles of 30 pixels each

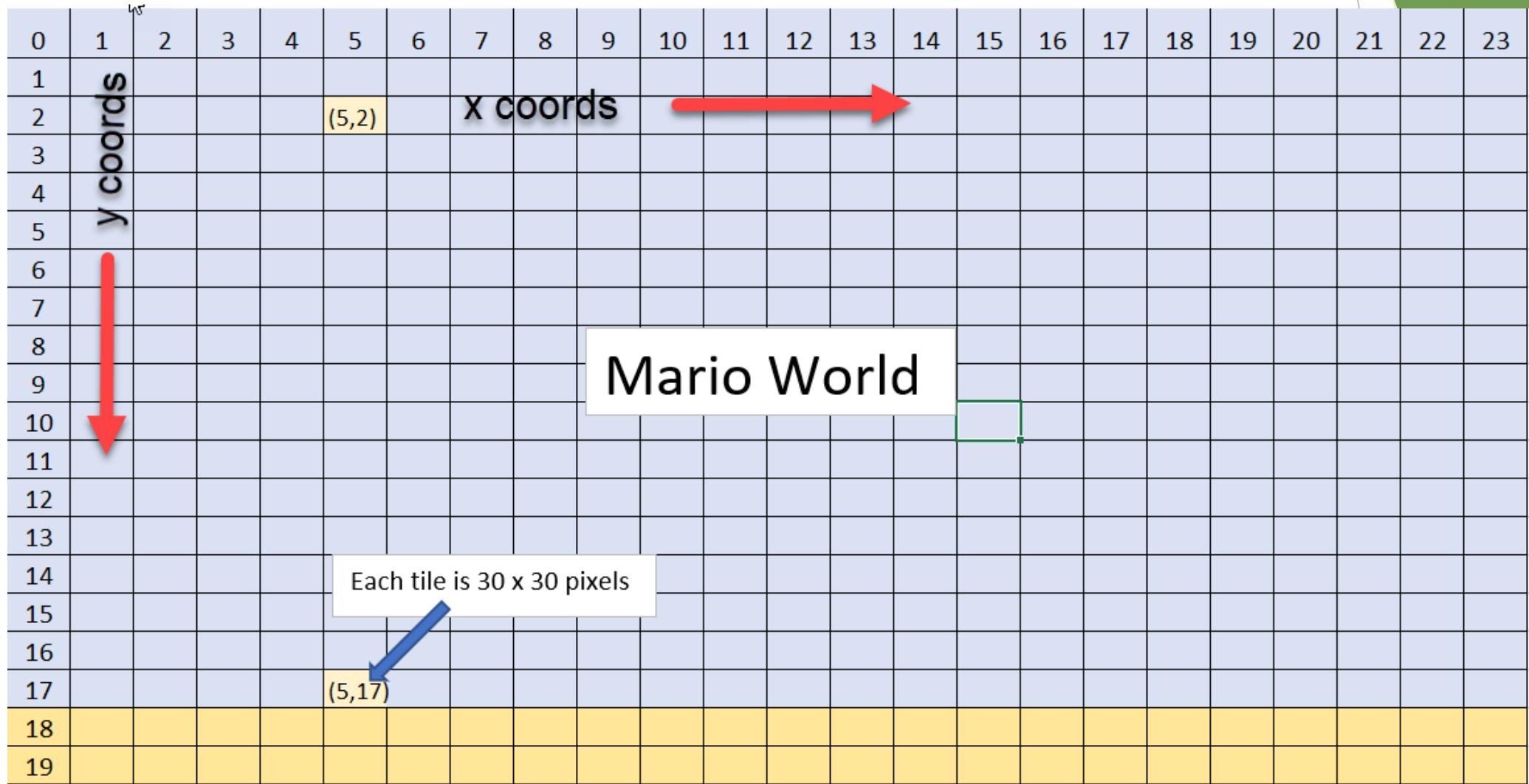
        super(MAXN_COLUMNS, MAXN_ROWS, TILE_SIZE);

        drawPath();

        mario = new Mario();
        addObject(mario, 1, GROUND_ROW);

        buildPyramid();
    }
}
```

2D Coordinates (x, y)



drawPath()

```
/*
 * Create a path at the bottom of the screen which is
 * 2 tiles high and goes right across the whole width of
 * the screen to form the ground for Mario to walk on.
 */
private void drawPath()
{
    int yStart = MAXN_ROWS - 1; // 19
    int yEnd = GROUND_ROW + 1; // 18

    for(int y = yStart; y >= yEnd; y--)
    {
        for(int x = 0; x < MAXN_COLUMNS; x++)
        {
            Block Block = new Block();
            addObject(Block, x, y);
        }
    }
}
```

Start at the bottom row of the screen and work upwards row by row and column by column

buildPyramid()

```
/**  
 * Build a pyramid of blocks. The pyramid base is twice  
 * the size, and the pyramid is size blocks high.  
 * There is a gap of 2 blocks in the centre  
 */  
public void buildPyramid()  
{  
    int size = getPyramidSize();  
    int x = 4; int y = GROUND_ROW;  
  
    Block Block = new Block();  
    addObject(Block, x, y);  
}
```

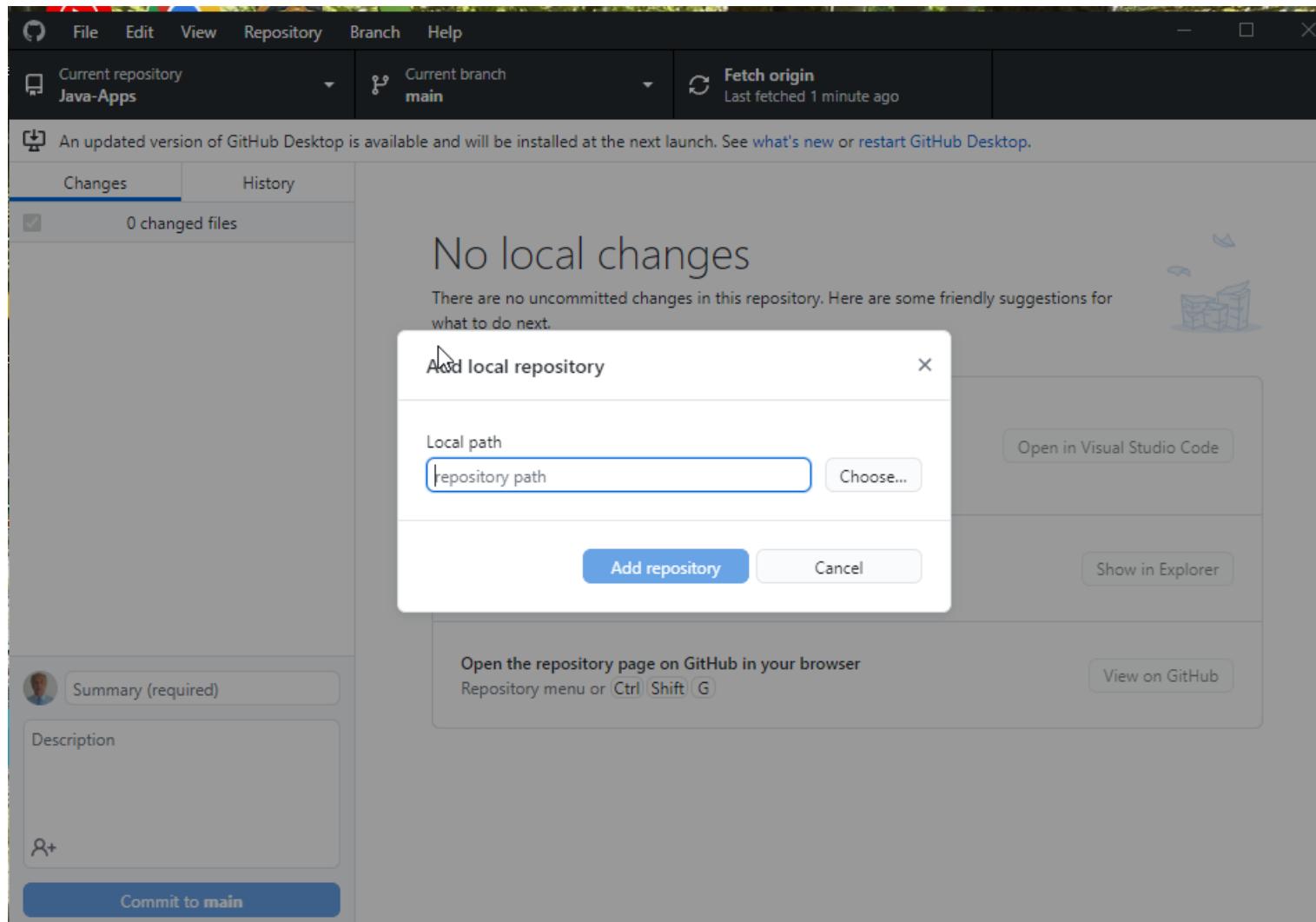
- Refactor getSize() so that it only returns valid values.
- Refactor build pyramid so that it can be built anywhere on the ground.
- Build one side of the pyramid
- Build the other side

```
/**  
 * Ask the user to enter the size of the pyramid in  
 * blocks between 1 to 8 inclusive  
 */  
private int getPyramidSize()  
{  
    String reply = Greenfoot.ask("Enter the pyramid size (1-8) > ");  
    int size = Integer.parseInt(reply);  
  
    return size;  
}
```

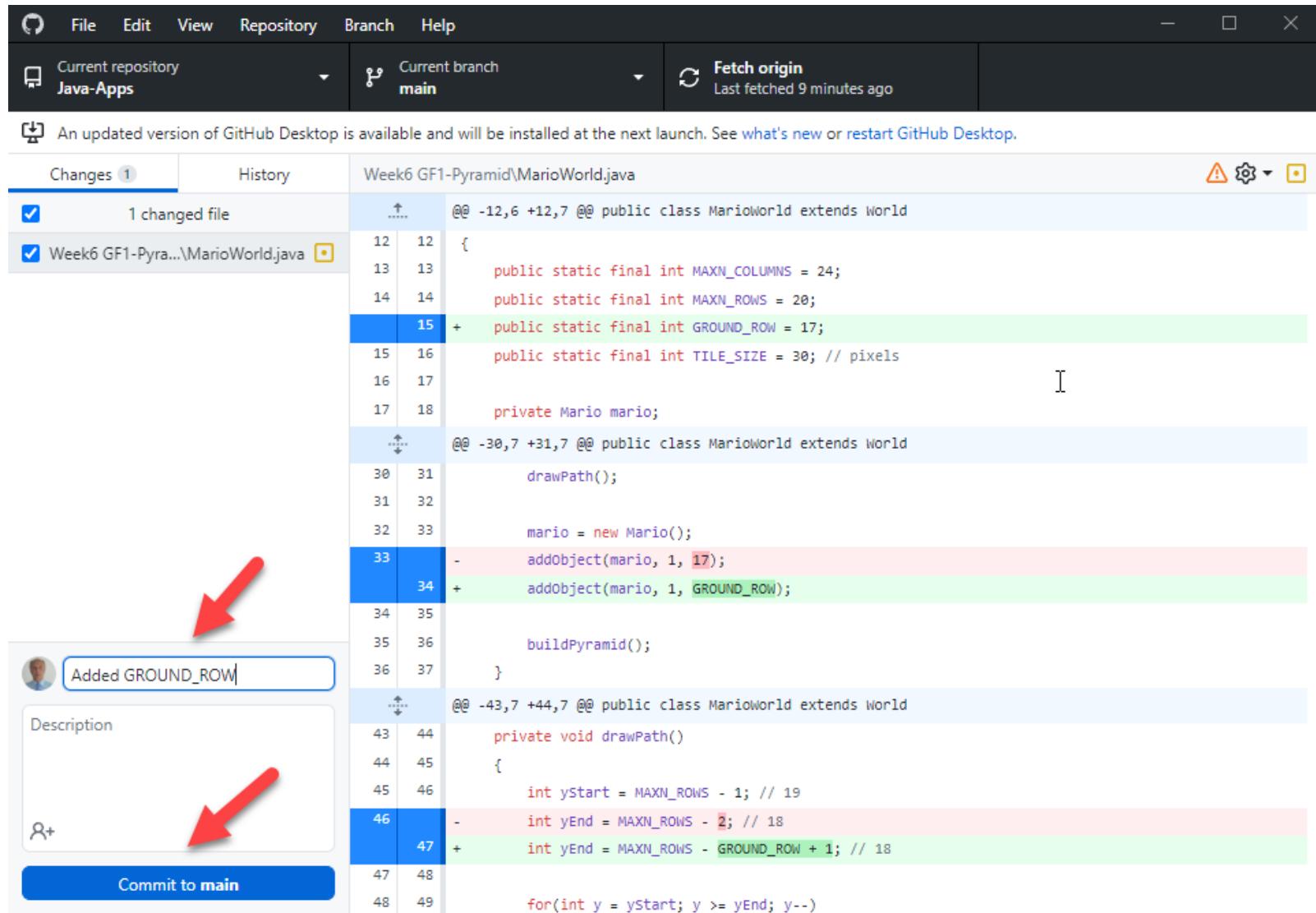
Using GitHub with Greenfoot

- ▶ Greenfoot does not know about Git or GitHub
- ▶ There are three Greenfoot Projects inside Java-Apps
- ▶ Use GitHub Desktop to save any changes to the Greenfoot projects
- ▶ Open the whole repository in GitHub Desktop

Add Repository to GitHub Desktop



Commit Changes to main



An updated version of GitHub Desktop is available and will be installed at the next launch. See [what's new](#) or [restart GitHub Desktop](#).

Changes 1	History	Week6 GF1-Pyramid\MarioWorld.java
<input checked="" type="checkbox"/> 1 changed file		@@ -12,6 +12,7 @@ public class MarioWorld extends World
<input checked="" type="checkbox"/> Week6 GF1-Pyra...\\MarioWorld.java		12 12 { 13 13 public static final int MAXN_COLUMNS = 24; 14 14 public static final int MAXN_ROWS = 20; 15 15 + public static final int GROUND_ROW = 17; 16 16 public static final int TILE_SIZE = 30; // pixels 17 18 private Mario mario; 18 19 @@ -30,7 +31,7 @@ public class MarioWorld extends World 30 31 drawPath(); 31 32 mario = new Mario(); 32 33 addObject(mario, 1, 17); 33 34 - addObject(mario, 1, GROUND_ROW); 34 35 buildPyramid(); 35 36 } 36 37 @@ -43,7 +44,7 @@ public class MarioWorld extends World 43 44 private void drawPath() 44 45 { 45 46 int yStart = MAXN_ROWS - 1; // 19 46 47 - int yEnd = MAXN_ROWS - 2; // 18 47 48 + int yEnd = MAXN_ROWS - GROUND_ROW + 1; // 18 48 49 for(int y = ystart; y >= yEnd; y--)

Added GROUND_ROW

Description

+ Commit to main

Main refers to the one and only git branch

Push origin

The screenshot shows the GitHub Desktop application window. At the top, the menu bar includes File, Edit, View, Repository, Branch, and Help. The repository dropdown shows "Current repository Java-Apps" and the branch dropdown shows "Current branch main". A status bar at the top right indicates "Push origin Last fetched 10 minutes ago". A notification at the top left informs the user about an available update.

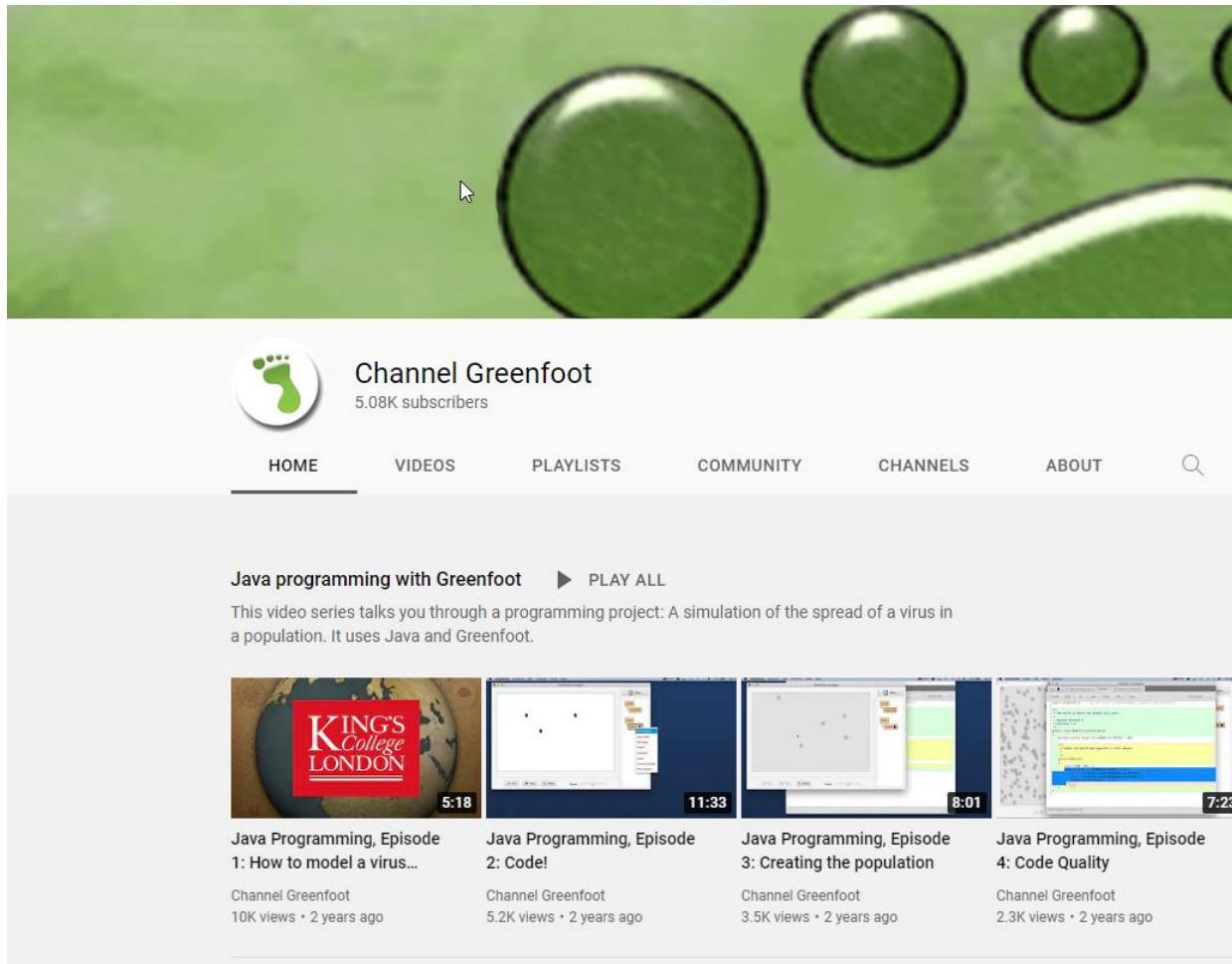
The main area displays a message "No local changes" with a sub-message: "There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next." Below this, a section titled "Push commits to the origin remote" shows "You have 1 local commit waiting to be pushed to GitHub." It includes a "Push origin" button, which is highlighted with a large red arrow pointing to it. A keyboard shortcut "Ctrl P" is also mentioned. To the left of this section, there's a summary card for a commit: "Summary (required)" with a user icon, "Commit to main" button, and a note "Committed just now Added GROUND_ROW".

Below the main suggestions, there are three other cards: "Open the repository in your external editor" (with "Open in Visual Studio Code" button), "View the files of your repository in Explorer" (with "Show in Explorer" button), and "Open the repository page on GitHub in your browser" (with "View on GitHub" button). The background of the application features a green abstract geometric pattern.

Greenfoot Videos

<https://www.greenfoot.org/doc>

<https://www.youtube.com/user/18km>



The screenshot shows the YouTube channel page for 'Channel Greenfoot'. The channel has 5.08K subscribers. The main video thumbnail is titled 'Java programming with Greenfoot' and shows a close-up of a green frog's face. Below the thumbnail, the video title is 'Java Programming, Episode 1: How to model a virus...' with a duration of 5:18. The channel navigation bar includes links for HOME, VIDEOS, PLAYLISTS, COMMUNITY, CHANNELS, and ABOUT.

Java programming with Greenfoot ► PLAY ALL

This video series talks you through a programming project: A simulation of the spread of a virus in a population. It uses Java and Greenfoot.

Video Title	Duration	Views	Last Updated
Java Programming, Episode 1: How to model a virus...	5:18	10K views	2 years ago
Java Programming, Episode 2: Code!	11:33	5.2K views	2 years ago
Java Programming, Episode 3: Creating the population	8:01	3.5K views	2 years ago
Java Programming, Episode 4: Code Quality	7:23	2.3K views	2 years ago

Summary

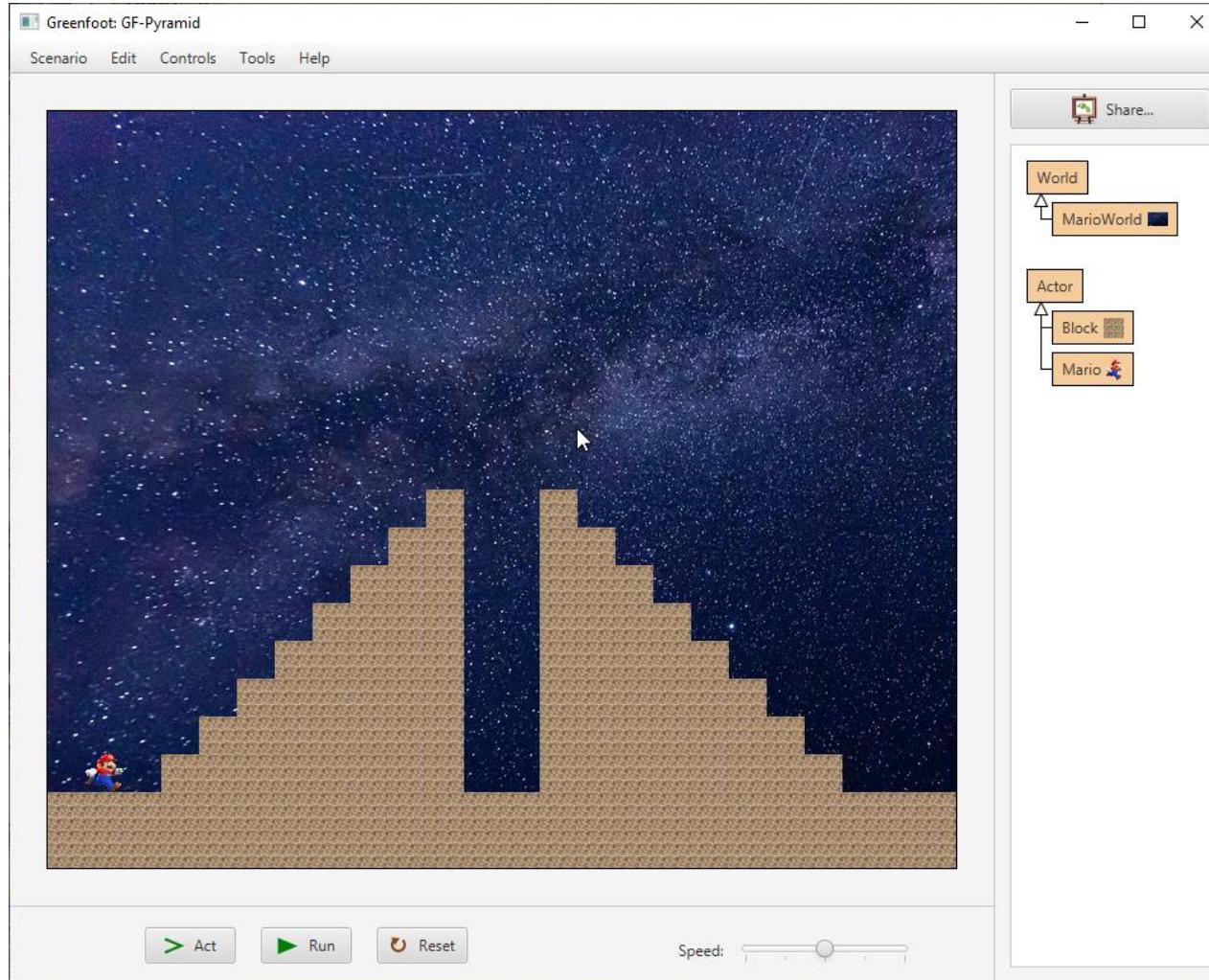
- ▶ Greenfoot can be used to create 2D Java games easily
- ▶ Greenfoot **Worlds** contain background images and **Actors**
- ▶ Actors can be created and placed in the World using (x, y) cords
- ▶ Actors will `act()` i.e. move around and do things.

Actor.act()

```
9 public class Mario extends Actor
10 {
11     private GreenfootImage image;
12
13     public Mario()
14     {
15         image = getImage();    [
16         int size = MarioWorld.TILE_SIZE;
17         image.scale(size, size);
18     }
19
20     public void act()
21     {
22         move(1);
23     }
24 }
```

- Mario is set to the same size as a tile
- When the app is run
- The act() method is called 30/sec.
- Mario moves 1 tile each time
- Mario goes straight through any blocks
- Intelligent movement and collision detection need adding.

Practical Exercises



- ▶ Draw a half pyramid of fixed size
- ▶ Change it so that it is of variable size
- ▶ Draw both half pyramids of variable size
- ▶ Can you get Mario to move??
- ▶ Can you get Mario to stop when he hits a block
- ▶ Where can you go to find out what other methods are available?