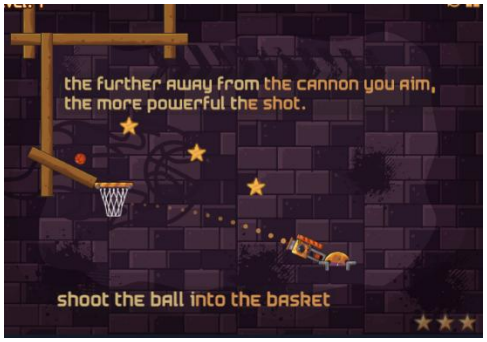
Analysis

My project will be a 2D projectile puzzle game called “PuzzleBall”. The gameplay loop of this game will consist of launching a projectile at adjustable power and angles to reach the desired destination and complete the puzzle. To make game challenging there will be a variety of obstacles in the way of the ball including basic walls and pitfalls with hopefully more interactive obstacles added if there is time, it will also include different materials of map to change friction.

The aim is to create a fun experience where players can visualise and play around with projectile mechanics. The user will be able to choose between puzzles I made to challenge themselves and experience the unique mechanics with increasing difficulty or a sandbox mode. In this sandbox mode the player with have the ability to place and move various objects from the game to do experiments or make their own levels. I believe this will be more used by students/teachers especially for experiments that may be hard to replicate in real life. This mode should have a variety of options for materials, objects and even the attributes of the ball such as weight, different materials could have unique effects (such as bouncy slime) or just varying friction.

Cannon basketball

Cannon basketball is a similar projectile-based game where you have to launch a basketball into a hoop whilst collecting stars. The simplicity allows players to pick up and play with success which is something I would aim for in my game, however I feel the lack of variation in obstacles and the



stars leading the successful shot creates a lack of creativity or learning to complete levels and improve.

Furthermore there seems to be a focus on an abundance of levels instead of a sandbox or challenge modes which I plan to add to my game.

The game manages to show substantial personality with its art style despite the basic premise. Whilst I do want my game to be aesthetically pleasing my priority will be on interesting mechanics which allow the player to explore unique interactions first.

Pros:

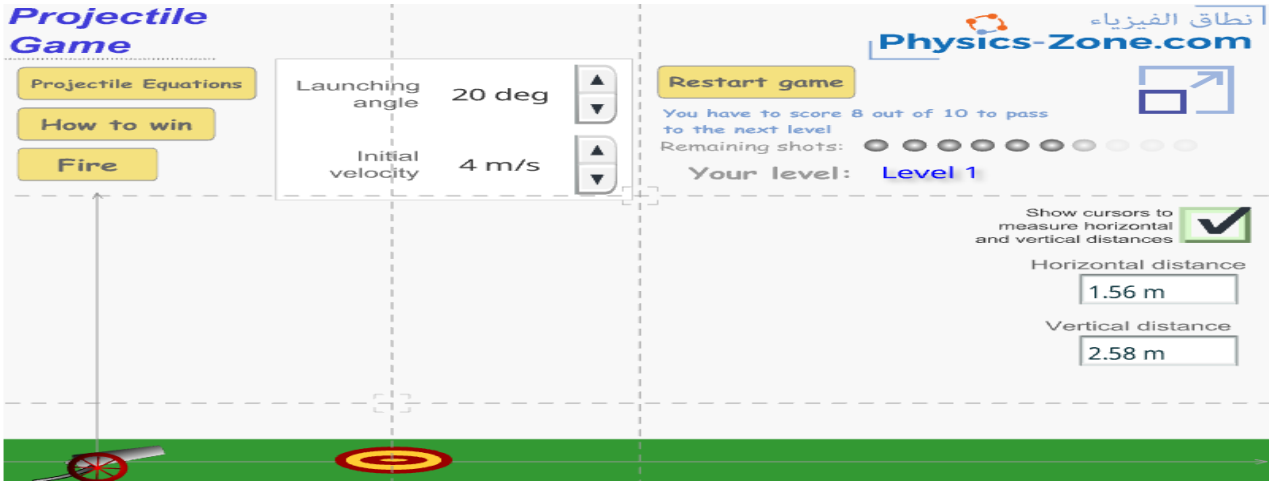
- Fun
- Stars add goals to collect
- Good art
- Large amount of levels to complete

Cons:

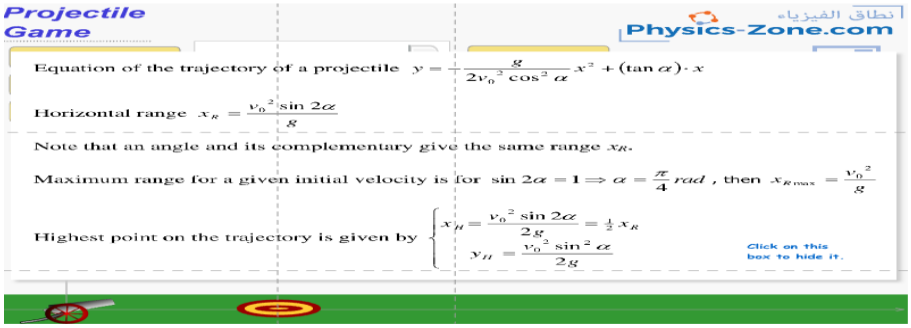
- Not many unique challenges
- Lack of difficulty(stars often guide your exact trajectory)

I think the stars are very interesting to consider, however for my project I don't think they are needed as they do not add much to the experimentation/ learning of this style of game and instead often end up just helping the player. This is also true for the trajectory predicter, It does add accessibility for younger players but can take away the thought from a more advanced physics student as it is easier to just aim and when combined with no shot limit can lead to "spamming". If I were to add a trajectory aimer I would have it be toggleable for it to only be used by players that need it.

Projectile game



This is a basic physics sim where you calculate each shot before you shoot. Being able to see exactly what the equations you use in mechanics look like in real life is very interesting it feels more like school than a game. This game also has very frustrating controls where to control the cannon you must click the arrows for each interval whereas I plan to use a slider for angle and power and displaying the exact values from it.



Pros:

- Displays equations used for projectiles, useful for teaching and understanding

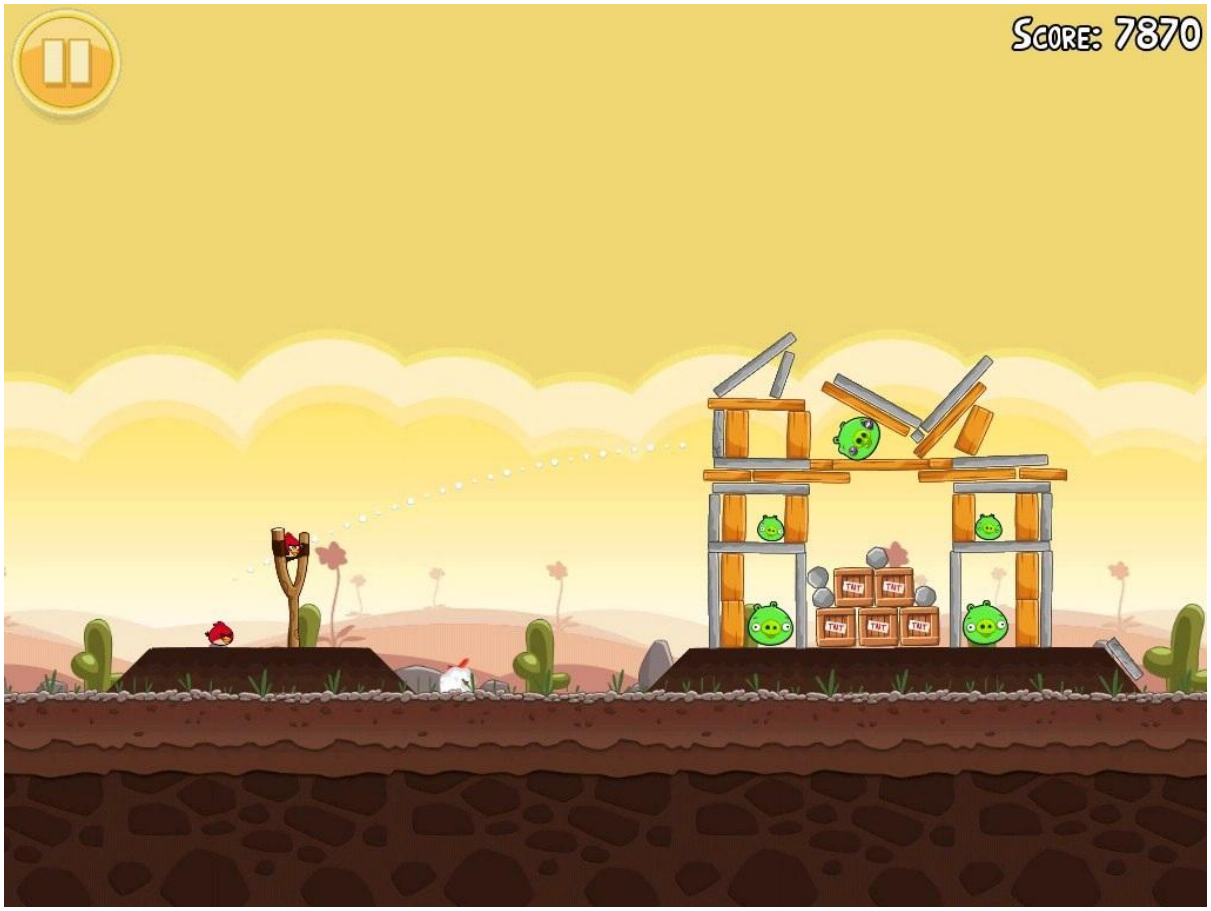
- Very simple, no visual clutter.
- The target randomly resets to challenge the player with different goals.

Cons:

- Not much replay ability as there is one “level”
- Equations are static and do not show what is being calculate specific to what the ball is doing
- No options to change the experiment
- Often ends with the user guessing instead of calculating exact projectile path

The equation display is a very interesting idea that I plan to use in my project. However, I would like an option to see current stats of the ball as all necessary information will already be stored in the code. I think a toggleable display to see acceleration, velocity etc of a ball can be very useful.

Angry bird



Angry birds is a very popular projectile game. It involves shooting birds at a building made smaller physics interactive objects and pigs you need to squash.

Pros:

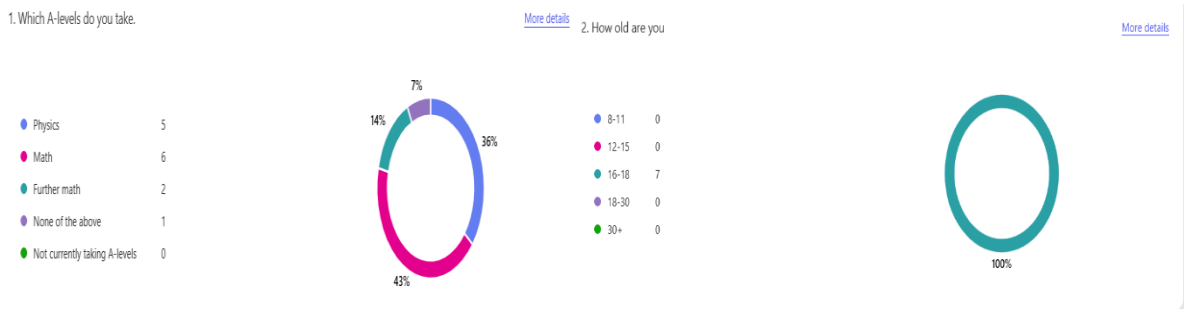
- Very recognisable and has unique personality
- Lots of levels
- Good variety in levels as there can be different objects to help or hinder the player (strong walls, explosives etc)
- A variety in the birds(projectiles) which can do different things, some can be strong, split into smaller parts or even explode.

Cons:

- Can be easy (but is understandable as aimed for a younger audience).
- Not good for experimentation

I think different projectiles is an incredibly unique and interesting mechanic but may be too much for my project. It would take significant development time and may not be as well suited to my game which will have intended solutions so could lack the space needed for this type of unique mechanics. Furthermore whilst the originality and character of Angry Birds is its' biggest strength, my project will be primarily focused on a range and polish of mechanics as it may be used as a tool.

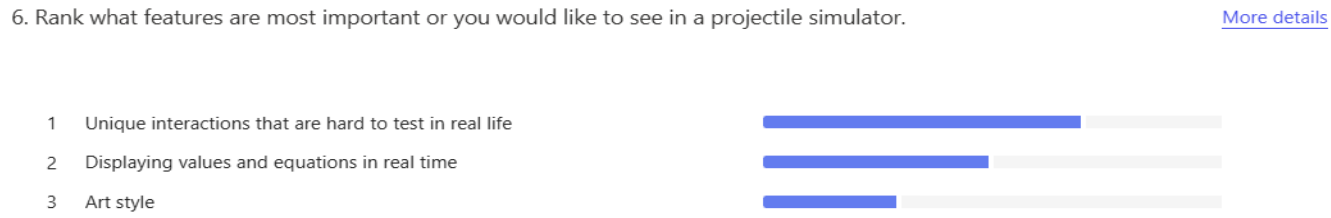
Stakeholders form.



All of my responders where at college age and a significant majority were taking physics and or math. This shows the potential userbase of my game and gives needed context to their other responses.

Most of my users would prefer a make your own mode in my game allowing them to do and test whatever they want with the mechanics in my game. This supported with the fact that a majority of people prefer to test unique interactions over all other significant possible features in my game shows that the main need of my stakeholders is a creative environment to learn

and test physics over hard and difficult levels.



How would you like forces on the ball to be displayed:

It was split mostly evenly between a status bar or vector arrows to display acceleration of the ball. I personally prefer a table/bar to display the forces as it will add a lot of visual clutter on the ball and could be hard to develop.

Requirements:

My priority requirements will be the physics system of my game. It needs to be able to shoot a projectile that can react with physical objects with different angles and speeds whilst looking smooth as this is the main function of the project. Secondly, It will need well thought out levels so there is something to play. However, I will only aim for three levels as making more can be very time consuming and will not add much to the user experience.

Feature		Sub-Feature	Explanation	Justification	Importance
Cannon	a	Adjustable angle	The player will need to be able to choose at what angle to shoot the projectile	Needed as one of two vital player inputs to affect the	Essential

				trajectory of the ball	
	b	Adjustable power	The player will need to be able to choose at what power to shoot the projectile	Affects how fast and far the ball goes so essential to solve levels	Essential
	c	See predicted trajectory	Ability to see a dotted line prediction of trajectory	May be helpful but could be difficult to develop	Medium priority
	d	Effects when shooting		My stakeholders did not care for graphics much but it would be easy to code	Low priority
Levels	a	Walls and floors	Stop the ball and will act as the main obstacles	Needed for there to be a game	Essential
	b	Goal	A target the projectile needs to hit. The target will have to be clearly marked so	Without this the levels will be incompletable and there will be no game .	Essential
	c	Ramps	Ramps/slopes in levels as obstacles or needed to speed up the ball	Will add a lot of complexity but will be difficult to code.	High priority
	d	Physics affected obstacles	Trap-doors, swings or other balls that will move when hit	Adds variety but will be difficult to make due to having their own	Medium priority

				physics and possible interactions with the ball.	
	e	Different material obstacles	Different types of walls/slopes etc. The default will be wood but there could be ice (very low friction), slime(bouncy) or sticky walls	Will add a large number of interactions for experimenting, make levels more interesting and harder but could be time consuming to add a larger amount so I will aim for three different types	High priority
	f	Amount of levels	I will add three levels to the game (easy, medium hard)	I think three levels of different difficulties will be enough to showcase many features. I will not add more as it will not change the experience but would take a lot of time I should spend developing other mechanics	Essential for three.

	g	Game modes	My game will contain both a level mode where you play through puzzles created by me and a sandbox mode.	I have decided to add two modes as my existing physics system can be used identically for both significantly reducing developme nt time if you were to do each individually whilst also greatly improving the experience. Most my stakeholder s do physics and have a high priority for testing different projectile interactions and this would allow that	High priority
GUI	a	Display of stats about the ball.	A toggleable display that will show attributes of the ball such as its horizontal and vertical speed/accelera tion.	Very simple to add as I would already have any needed information in the code.	High priority
	b	Restart level	Ability to reset the ball back in the cannon and try again	Needed otherwise you would have to	Essential

				complete each level first try	
	c	Level select screen	The user should be able to select levels without having to do them in order. They should have the option to pick between the four levels being easy, medium, hard and sandbox	Should not be harder to implement than the alternative of doing the levels linearly but I would prefer this as players may be varying skill levels.	High priority

Limitations:

Art style – I would like to have a unique and polished look for my game however creating good sprites, backgrounds and effects would take a large amount of time which should be spent on the game's mechanics. My objects and ball will just be block colours as to reduce visual clutter whilst still be able to tell objects apart from each other.

Music/sound effects – I personally believe these add a lot to the experience and personality of a game but I have never developed music before so it would a huge time sink. Considering my limited time, I see no reason to prioritise this over other features.

Number of levels – I aim to have three levels to my game and a sandbox mode. I do not want to waste time developing many unique levels, instead creating three thought out ones. I believe that this will be enough to experience the game and have progressing difficulty but if I had more time adding more levels would be a priority.

Saving levels – In my game there will be a sandbox mode where the user can create their own levels for experimentation. I do not plan to add an ability to save created levels for later use as it will be challenging to develop and it should not diminish users experiences significantly. If I do have time and finish other higher priority tasks I could add this feature with the help of this https://docs.oracle.com/javase/tutorial/essential/io/file.html documentation

Hardware requirements:

- Due to my project being a relatively simple game with not much having to be processed at once it should be able to be run on any computer.
- It will require a mouse instead of a controller but as it is only navigated with clicks you should be able to play without a keyboard.

Software requirements:

- To access the game the user would either have to download green foot and the game files or access it on the webpage.
- Both these solutions require internet access however you will only need it once to download the game if you do are not using the website.

Computational methods:

Physics simulation is very well suited for computers compared to human calculations as they can run calculations very quickly and concurrently in the case of multiple physics-based objects. Furthermore, my project will allow the visualisation of these problems along with the option to display information.

Abstraction will be used as the user will be presented with only necessary information, everything on the screen should be used in the puzzle. From the stakeholders' responses I can see art is not important for this project so unneeded props or decorations will not be added. This will reduce development time and clear the screen for users to better visualise their experiments.

Development plan:

Cannon:

The cannon will be where a majority of user interaction takes place. It is present in every level and is also very important to run smoothly for user enjoyability. I plan to be able to complete this in one-two sessions as it should be quite basic. It will need:

- Power and angle slider
- Optional text input for both
- Follows play cursor
- Fires when button is pressed
- (Optional) Trajectory predictor

Test No.	What is being tested	Type of test	Expected	Result
1.	Fire button	Valid	Ball is projected out of cannon	
2.	Power slider	Valid	Balls power is increased with slider	
3.	Slider stays contact	Invalid	Slider pointer doesn’t follow the mouse past the end of the slider	
4.	Angle	Valid	The angle of cannon and	

			the projectile is correctly	
5.	Trajectory predictor	Valid	Curved line follows cursor that adjusts with the parameters.	

Physics system:

I plan to develop the physics system first as it is the base of the whole project and needs to well polished for it to run smoothly. Whilst I have experience coding basic gravity I have never made many of the other essential functions so I predict the physics system to be my longest stage of development. It will include:

- Gravity
- Air resistance
- Friction
- Slopes
- Momentum when colliding with other physics based objects

Test No.	What is being tested	Type of test	Expected	Result
1.	Collision	Valid	Projectile doesn't get stuck in or “phase” through other objects	
2.	Gravity	Valid	Projectile accelerates down at expected rate	
3.	Friction	Valid	Projectile slows down faster on certain objects	
4.	Momentum	Valid	When one physics	

			based object hits another they transfer the forces correctly.	

Stages:

In the “levels” mode the user will play through levels of varying difficulty of obstacles that get in the way of the ball reaching a set goal. I only plan to make 3 as It may take a lot of time. They will need:

- Goal
- Complete/Next level screen
- Walls, Slopes and ramps
- Other physics based objects(such as trapdoors, swings and other balls)
- Different materials of obstacles with special effects.

Test No.	What is being tested	Type of test	Expected	Result
1.	Goals/ complete screen	Valid	When projectile reaches goal, level ends with complete screen	
2.	Walls	Valid	Walls stay anchored in place and collide with other objects.	
3.	Slopes/ramps	Valid	Slopes and ramps slow down or speed up the ball.	

4.	Trapdoors and swings			
5.	Bouncy slime, bounciness		Ball bounces more when falling on slime compared to other objects	
6.	Ice, friction		Ice slows down the	

Sandbox mode:

The sandbox mode will allow users to test physics scenarios inside the game or make their own fun levels. As it is using pre-existing code for the physics system and obstacles the main development in this stage will be a menu and feature the choose/places objects in the world. I do have limited experience with player adding objects into the world however without a grid it may be difficult to stop collision of objects.

- Menu to select objects
- Place objects on cursor

Test No.	What is being tested	Type of test	Expected	Result
1.	Place object	Valid	Object is placed on cursor after selected	
2.	Can't place object inside other	Invalid	Does not allow you to place object where another one is located	
3.	Select correct object	Valid	Correct object is selected	
4.	Drop down menu	Valid	Menu appears	

			when clicked.	
5.				
6.				

Debugging/Finish Code:

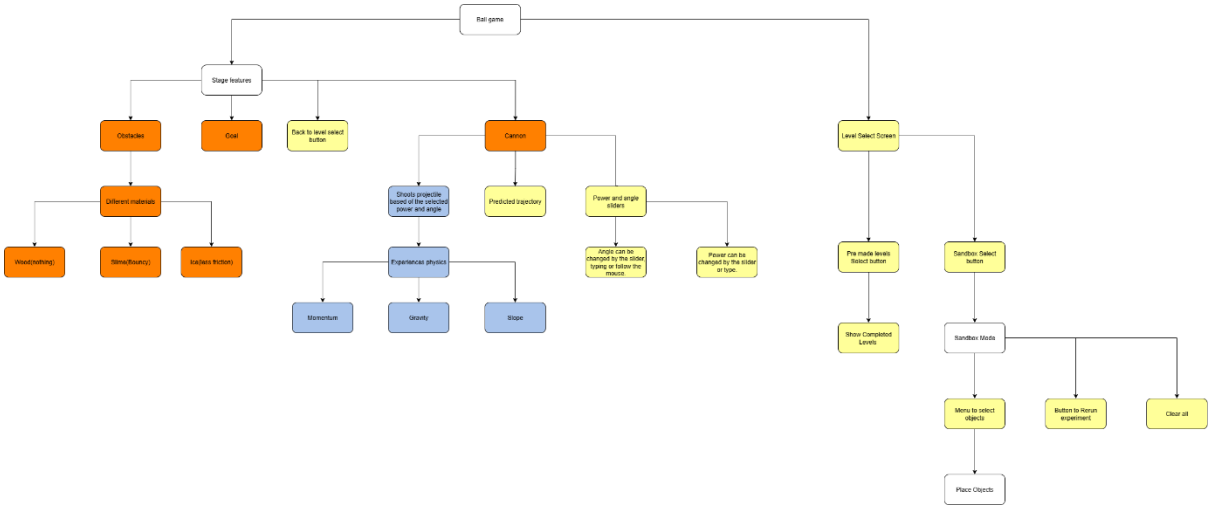
Although I will debug and test during development I plan to have at least a week to thoroughly check all parts of the project work as intended.

Structure diagram:

Brown - Objects in the stage

Yellow – Gui

Blue - Physics

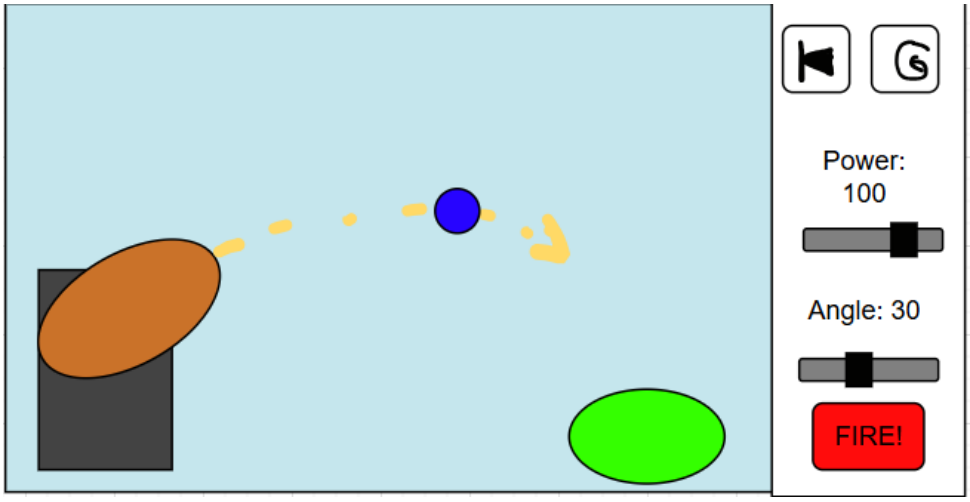


I chose to have cannon, Stage features and the select screen as my three distinct main groups because they handle the main aspects of the game with little overlap.

The cannon contains nearly all the player input and is where the projectile is fired from whereas the stage features group contains all the physical objects in the levels and their effects. Level selection is completely separate from the actually gameplay as it is entirely Gui and menus.

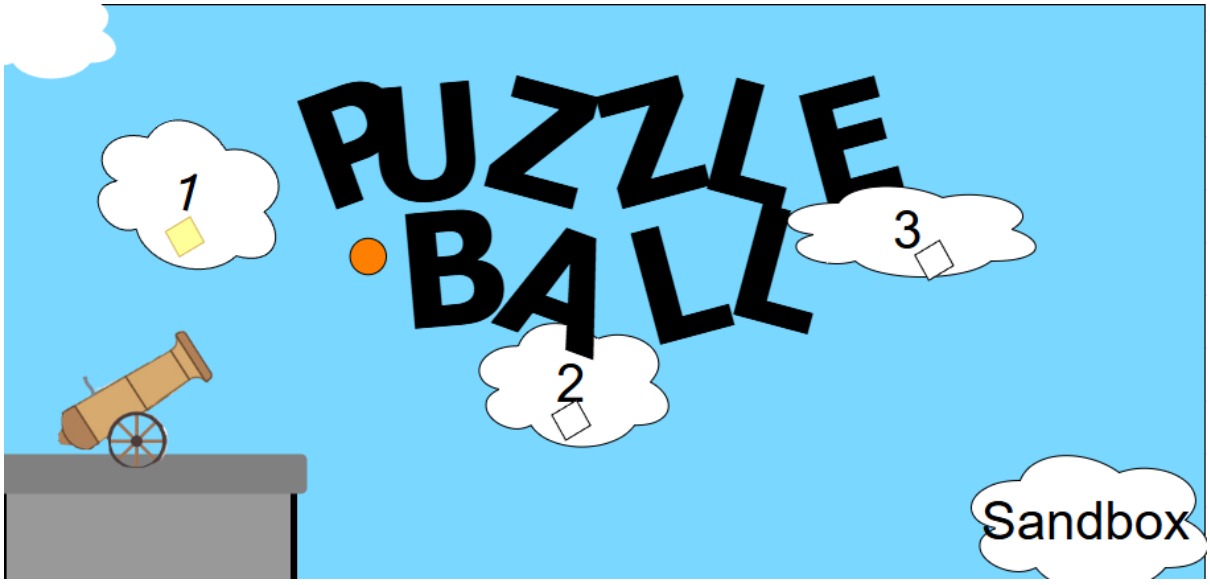
GUI:

The colour palette is somewhat muted as to support the simplicity of my project and keep everything clear, the ball and goal however stand out as much more powerful colours to show their importance and keep attention.



All player input is done on the right hand of the screen to keep the simulation clear. Both the power and angle can be adjusted by either dragging the slider or editing the number representation with the option you don't pick adjusting alongside. In the top right there are the back and restart buttons represented with commonly used icons to reduce word clutter whilst still being intuitive and easy to understand. The last of the buttons is the fire button which is marked in bright red to stand out against the red.

Level select screen:

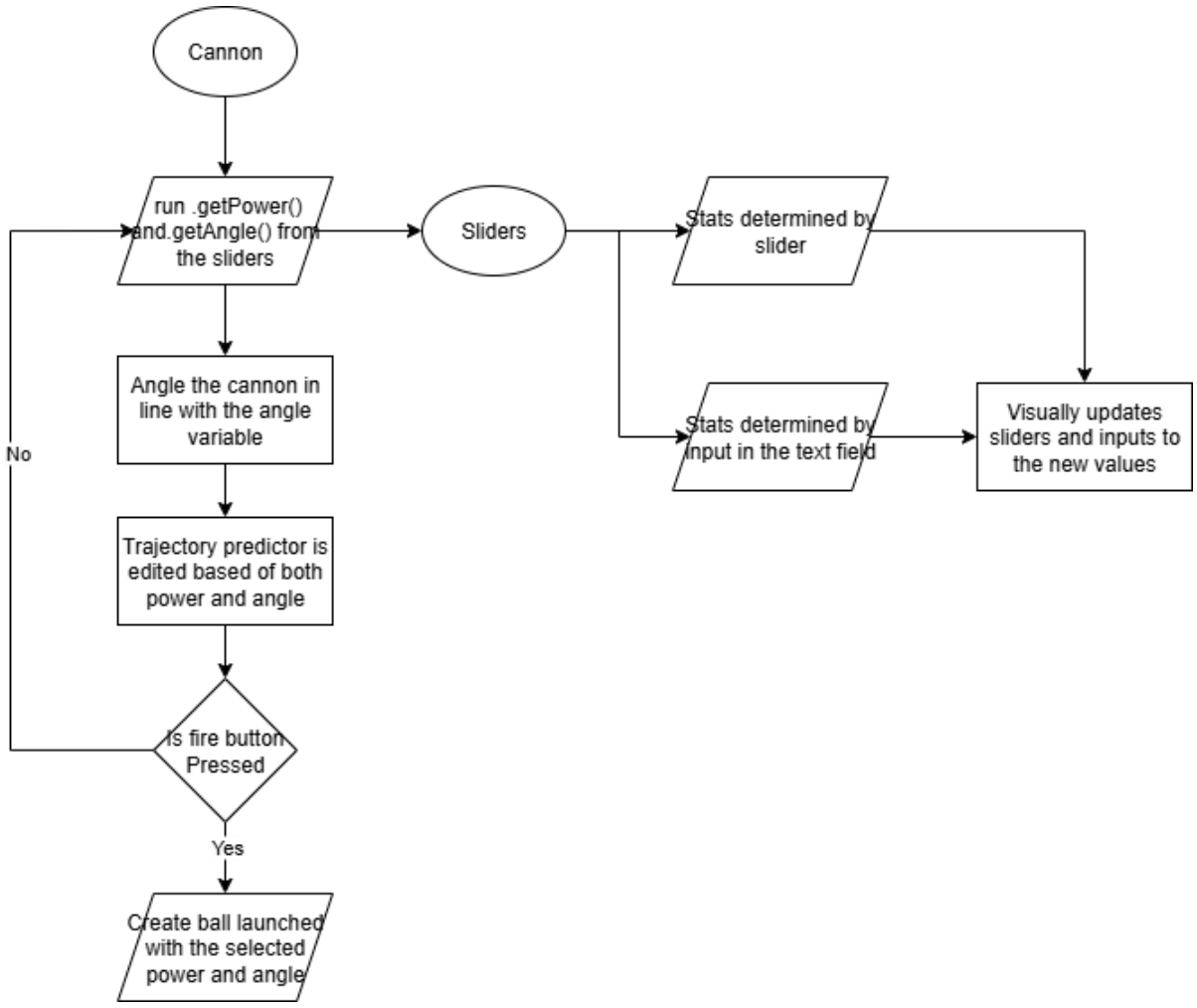


I decided to combine both the title screen and level select screen in one as I felt having two screens was unnecessary. The level buttons still blend into the theme but the outline with level number shows that they are interactive. The diamonds on each level show if it has been completed, where it is filled in with bright yellow if it has.

There is minimal reading and it is intuitive so anyone should be able to navigate comfortably.

Stage 1 Cannon/Sliders:

This is the first stage as without the cannon and the sliders to control it there is no input to the simulation. They also are some of the only features that do not rely on other systems being finished first to use effectively.



Pseudocode for power slider

```
//instantiate variables
Int power = 0;
Int angle = 0;
//Creates slider (exact values are yet to be decided)
fillRect(getX(), getY(), width, height, BLUE);
//Create visual button for the slider
sliderVisual button = sliderVisual
greenfoot.getWorld().addObject(button, getX(), getY())
Act(){
        Mouse mouse = Greenfoot.mouseinfo();
        updateVisualPos ();
        sliderInput();
        textInput();
}
sliderInput(){
        //is the mouse within the bounds of the slider
        If((mouse.getX()>getX() && mouse.getX()<(getx()+length)) &&
        mouse.getY()>getY()&&mouse.getY()<getY()+height(){
                //Check if the mouse has been clicked
                If(mouseClicked()){
                        //Total power is a placeholder until I decide the actual limits
                         Power = (mouse.getX() / getX()+length) * totalPower
                }
        }
```

```
}


updateVisualPos(){
        //The button on the circle needs to be updated along to match up with the power
        variable, along with the other way round
        sliderVisual.setPosition(power/totalPower*length + getX(), getY());
}
textInput(){


}
```

I decided to make the button of the slider purely visual as to allow the user to click anywhere of the slider to update the values and it does not require me to track when and where the mouse is held down.

Data dictionary:

Name:	What class is it stored in	Data Type	Description
angle	aSlider	integer	Stores the angle that the ball will be fired. Called with .getAngle();
power	pSlider	integer	Stored the power that the ball will be fired at. Called with .getPower();

Sliders:

First I had to create each part of the slider with Greenfoots draw functions.

Which I needed to reference from the Greenfoot API. (Greenfoot, 2025)

```
public Slider(){
    //Create slider
    GreenfootImage rect = new GreenfootImage(width,height);
    setImage(rect);
    rect.setColor(Color.BLACK);
    rect.drawRect(0,0,width-1,height-1);
    rect.setColor(Color.GRAY);
    rect.fillRect(1,1, width, height);

}
```

“Head” of the slider

```
GreenfootImage rect = new GreenfootImage(width,height);
setImage(rect);
rect.setColor(Color.BLACK);
rect.drawRect(0,0,width,height);
rect.setColor(Color.RED);
rect.fillRect(1,1, width, height-1);
```

Text

```
MyWorld world = (MyWorld)getWorld();
power= world.getSlider().getPower();
GreenfootImage text = new GreenfootImage(65,30
setImage(text);
text.setColor(Color.BLACK);
text.drawString("Power: "+power, 0, 10);
```

To combine them I chose to set their location relative to the slider instead of manually as it allows the flexibility to move the slider and easily have the others follow

```
ublic void setOthers(){
    MyWorld world = (MyWorld)getWorld();
    world.getpText().setLocation(this.getX()+10,this.getY()-20);
    world.getpHead().setLocation(this.getX(),this.getY());
}
```

When coding the sliders I need each component to interact with each other for calling functions. However I ran into some issues

```
ava.lang.NullPointerException: Cannot invoke "MyWorld.getSlider()" because "world" is null
```

```
public pText(){
    MyWorld world = (MyWorld)getWorld();
    power= world.getSlider().getPower();
```

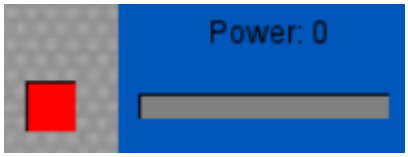
This was because it was in the constructor method so was called before the object existed in the world causing it to return null when finding what world it was in.

Next step was to make it functional. To start this I wanted to check when the mouse was in bounds of the slider, however this caused Greenfoot to instantly pause with no error code. This was caused by trying to run code on a null mouse object. To fix this I added this line

```
if(mouse != null){
```

Now I had to make the head follow the mouse when dragged. I used this code but there were no limits on where it would follow to.

```
if(Greenfoot.mousePressed(this)==true){
    mouseDown=true;
}
else if(Greenfoot.mouseClicked(this)){
    mouseDown=false;
}
if(mouseDown){
    setLocation(mouse.getX(),getY());
}
```



This was fixed by adding upper and lower limits to the y and x value of the mouse that counts as being on the head.

```
if(mouse!=null){
    //Check within Y bounds
    if(this.getY()>mouse.getY()+10 || this.getY()<mouse.getY()-10){
        mouseDown=false;
    }
    //Check within X bounds
    if(mouse.getX()>slider.getX()+50 || mouse.getX()<slider.getX()-50){
        mouseDown=false;
    }
}
```

With this equation I can now get the power from the slider

```
//Power equation(placeholder)
power = head.getX()-468;
```

Cannon

Firstly, I need to be able to access the stats from the angle and power sliders. Which is done simply by adding get functions in the needed actors.

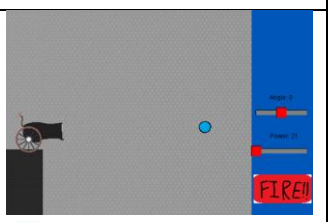
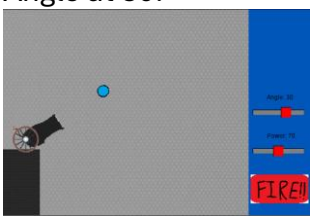
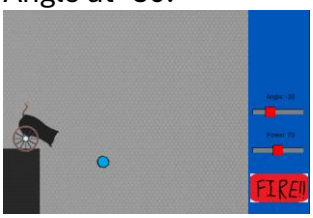
```
//Get needed objects
MyWorld world = (MyWorld)getWorld();
aSlider aSlider = world.getaSlider();
pSlider pSlider = world.getpSlider();
//Set angle of the cannon
int angle = aSlider.getAngle();
setRotation(-angle);
//Set power
int power = pSlider.getPower();
```

To fire the ball I used this

```
public void fire(){
    MyWorld world = (MyWorld)getWorld();
    world.addObject(world.getBall(), this.getX(), this.getY());
    world.getBall().setRotation(getRotation());
}
```

However this ended up with the inability to fire a second ball and only change the direction.

Test:

Test No.	What is being tested	Type of test	Expected	Result
1.	Fire button	Valid	Ball is projected out of cannon	 ✅
2.	Power slider	Valid	Balls power is increased with slider	Proof in video
3.	Slider stays contact	Invalid	Slider pointer doesn't follow the mouse past the end of the slider	Proof in video
4.	Angle	Valid	The angle of cannon and the projectile is correctly	Angle at 30:  Angle at -30:  ✅