#1GAM October - Candy

# Game premise

The idea behind this project is to make a short, physics-based game where the player-controlled character throws candy at the mouths of children with a slingshot “in style”. Depending on the kind of candy thrown, the physics will behave differently for each type, as specified a bit further in the document. It’s entirely a puzzle game, rather than an action-packed game. Levels will not have a time limit, but a “move limit”. There will be “X” candies to throw at “Y” Kids, and this number might not match. For you to understand how the game should work, think of a hybrid of “Angry Birds” (Rovio) mixed up with “Worms” (Team17), throwing a factor of “The incredible machine” (Sierra) to make the puzzles solvable.

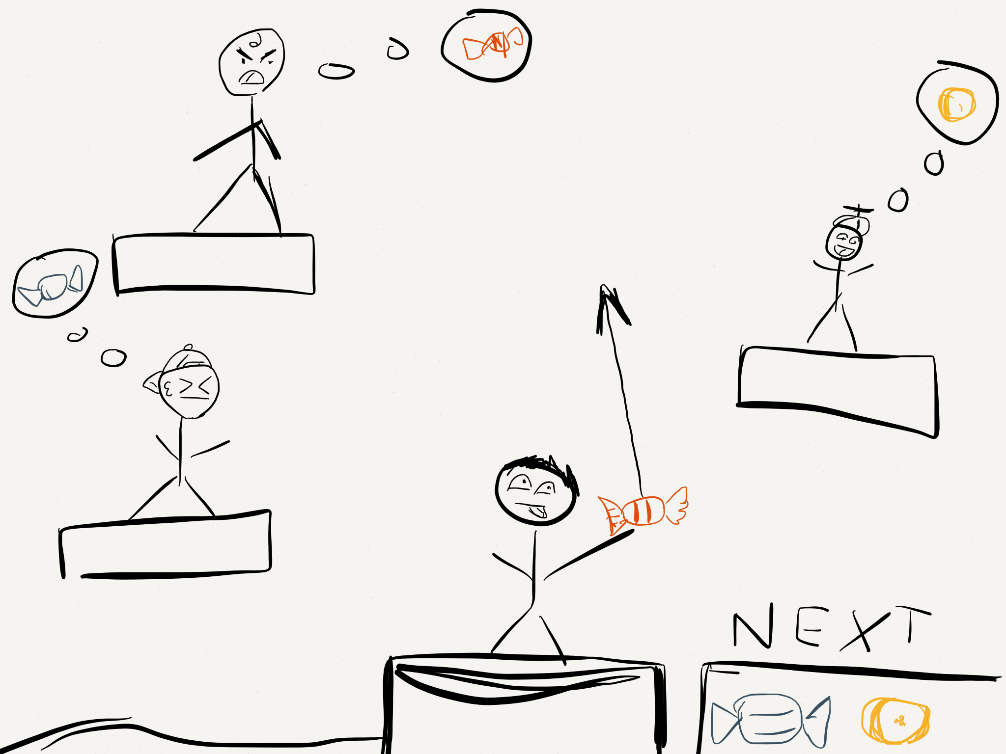
The game will have a 2-dimensional setup, and the graphic style is aimed for cartoonish characters, or pixel art (depends on the artist). It will be developed on Unity, and it is oriented to be played on the browser through Unityplayer. It can also be ported to Tablets and Phones.

If possible, the game should have an easy way for the Game Designers to create levels through Unity to relieve coding efforts. Detailed instructions on how the levels should be designed will be discussed a bit further in the document.

# Main Game Mechanics

Each of the levels will have a delimited area of interaction for the player. The main character will be placed on one spot on the map and won’t be able to be moved. This position will be determined by the level structure to match the challenge of the level. There is virtually no limitation to where the main character can be positioned at the beginning of the level.

Basically, a standard level of the game should look something like this:

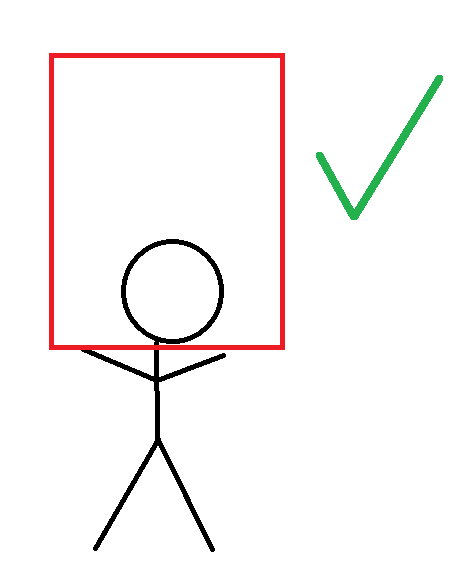


## Common level mechanics

Every level will have the same determined factors at the beginning of the level (Fixed factors)

* Position of the Main Character.
* Position of the Goal Characters (kids wanting candy).
* Candy that the main character can throw, and which kind. (But not the final quantity!)
* Level layout (Positioning of platforms, walls, and such).

The goal of each level is to throw the candy that each one of the kids desires to their mouths directly. To achieve this purpose, the main character will have to take into account some physic factors such as gravity, wind, or the kind of candy in order to achieve its purpose. There are, however, some basic rules that cannot be broken.

* The kids will **only** take the candy they want, given by a bubble over their heads. This bubbles will appear once when the level starts, and will appear again if the player hovers the cursor over the kids.
* The player does **not have to** aim exactly where their mouths are. While it always depends on the kid, there is usually a slight error margin. While placing the candy directly on their mouth will give out the max score, if the player throws the candy slightly over the kid’s head, the kid will jump and grab it, given that it is the candy he wanted. To understand this mechanic better, look at the following image:  
  

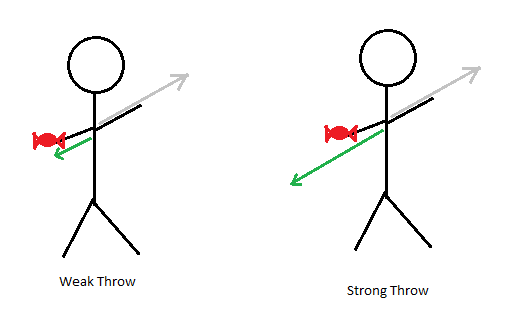
If the player tosses the candy on the area represented by the red square, the kid will jump and catch the candy. If he misses this area, or if he throws the wrong candy, he won’t move at all. This effect varies from kid to kid (see “CHARACTERS”).

For maximum score, the candy must go straight to the kid’s face (straight or in a parabola).

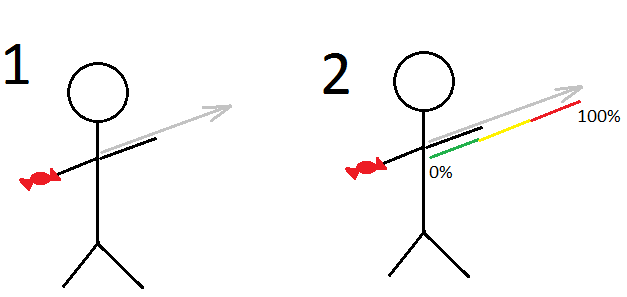
* Throwing a candy to a kid with too much strength to the face will not make him eat it, but for him to get smacked and injured, which will result as a **level failure** at the end of the level. The player has to be careful not to use too much force upon throwing.
* **No kid** will try to eat a stone. Throwing a stone at a kid will result in injury and level failure.
* The level ends when all kids have eaten their candies (the player wins), or when the player is out of things to throw, and there’s still at least one kid without candy (the player loses).
* Physical factors given at the beginning of the level, such as gravity force or wind strength will be set at the beginning of the level and will **not change** while the level lasts.

## Controls

There are two kinds of control we can choose for this game to develop on: Mouse and Keyboard (or both, if we want to give the choice to the player!).

* **Mouse controls:** To throw a candy the player clicks on the main character, and drags the cursor to the opposite direction to where the candy will be thrown. The force of the throw is given by how far away from the main character the player drags the mouse. The direction to where the candy will go will be represented by an arrow on the opposite direction (as well as the position of the main character’s other arm), but it will not be affected to the force applied. For example:   
  ****  
  As soon as the player lets go of the button, the candy will be thrown to the direction and strength the player gave it. There is a limit of the strength the throw can have, regardless of how far he holds the arrow back.  
    
  **Pros of using this control scheme:** 
  + It is very accurate.
  + It can be easily ported to tablets and mobile.

**Cons of using this control scheme:**

* + It is prone to mistakes of accidentally letting go the mouse and wasting a throw.
  + It is harder to calculate the strength of the throws.
* **Keyboard controls:** To throw a candy, the player first selects the direction they want to throw the candy with the “up” and “down” arrow keys (or W and S for the FPS-type control). The direction will be displayed by an arrow, and by the other arm of the main character, as well (just like the mouse controls) Once the player has selected the direction he wants to throw the candy to, he holds the throw button (space bar) to decide the strength of the throw. A bar will appear below the direction arrow to decide the strength of the throw, as explained by the following image.  
  ****The player will stay on position 1 for as long as he needs to calculate his throw. Only when he presses the throw button the position 2 will start. The power bar will start to fill gradually from 0 to 100 in an interval of two seconds. Upon hitting 100%, if the player has not let the button go, it will go backwards from 100 to 0. If the player has not let go either way when it comes back to 0, it will cancel the throw entirely and the player will have to press the button again in order to throw it again. Having pressed the throw button does **not lock** the player from correcting the trajectory of the candy with up and down should he desire to. The moment the player releases the throw button the candy will be released from the main character’s hands with the trajectory and strength provided upon the button release.  
    
  **Pros of using this control scheme:**
  + It’s a lot more accessible than mouse controls.
  + Players can be much more thorough and technical upon calculating their throws.
  + It can be a lot more accurate than the mouse control once the player gets used to it.

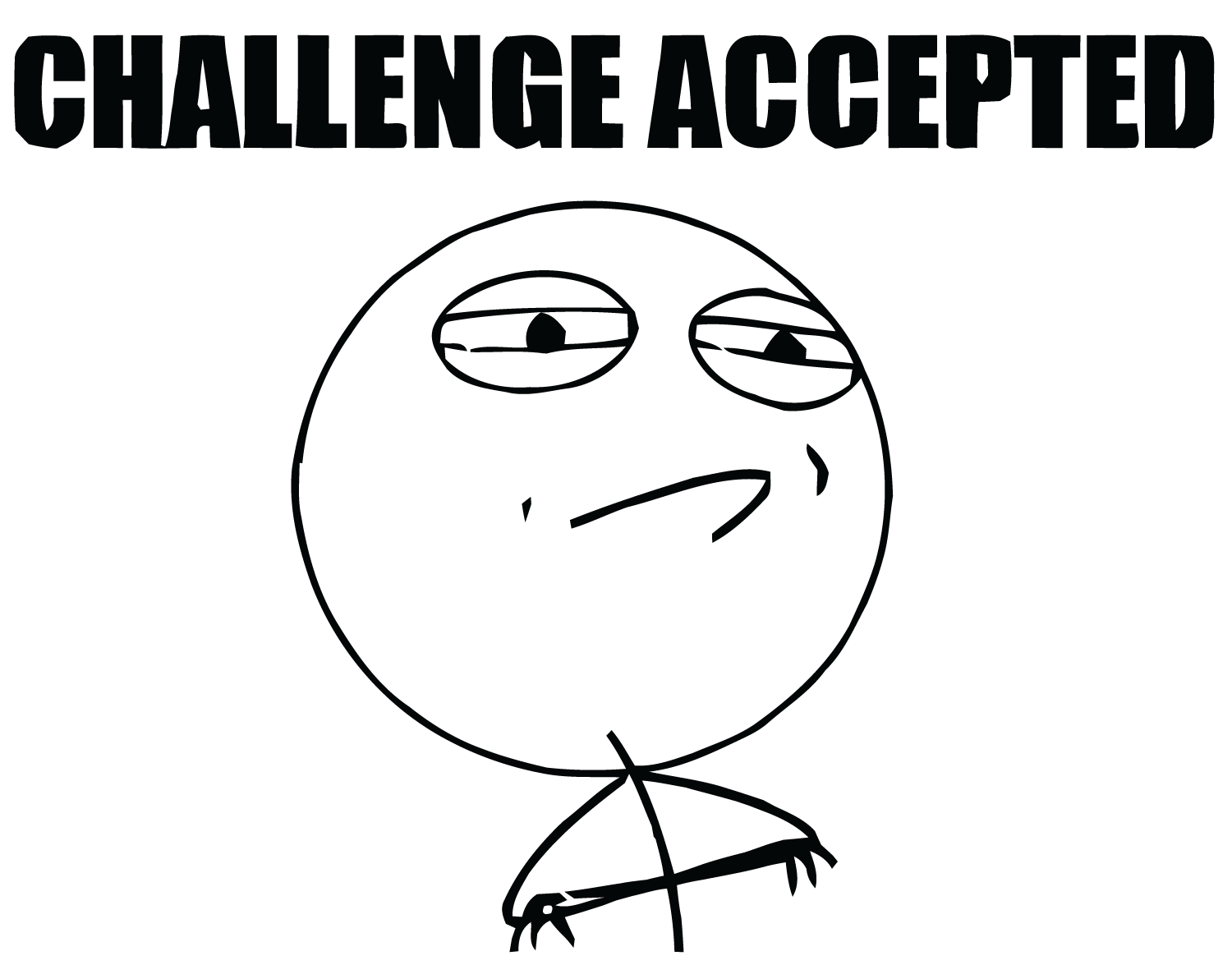
**Cons of using this control scheme**

* + It would limit the game to be PC/Console only.
  + Newcomers and tablet players might not find it so easy to understand.
  + It would very likely be much more difficult to implement than mouse controls.

# Characters

There are very few characters on (game name) but each one has their own personality and should be reflected into the game’s style.

## Main Character (The player names it?)

**Age**: 15

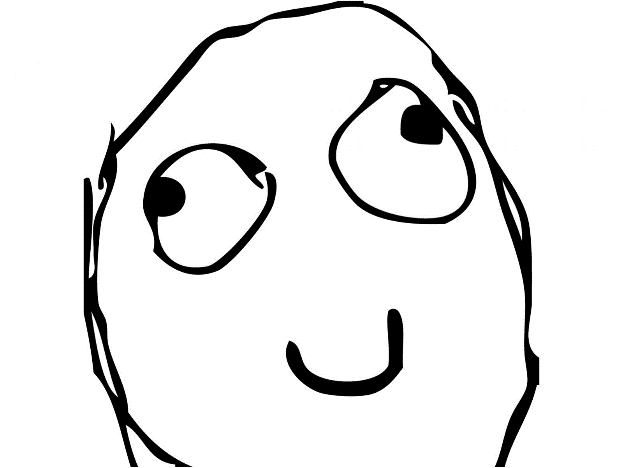
Aims to be a professional darts player. He is already a

renowned player on his town and is always on the look for a new challenge. He usually wears jeans, a cap, and a jacket (although the appearance can be changed with in-game unlocks –TO BE DECIDED-).

He’s a kind person, and always thinks of the others before himself. He lives with his family (mother, father and little brother).

His great imagination led him to invent this particular sport to hone up his skills, face new challenges and make everyone happy around him.

## Mike



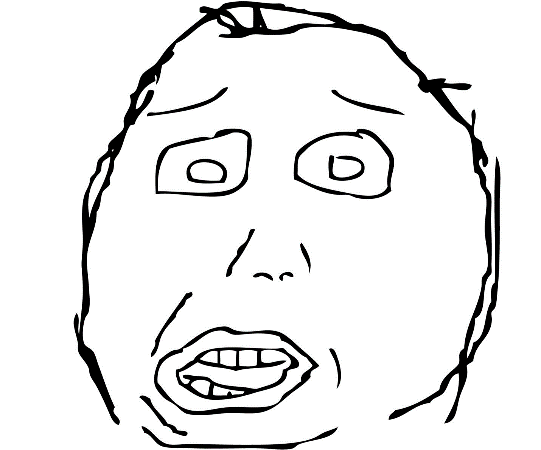
**Age**: 9

**Jump distance:** Medium

**Favorite candy:** Gum

Mike is pretty much every kind of definition of an “Average Joe”. He lives a common life, gets common marks at school, has the normal child hobbies, and loves any kind of candy in equal measure. He usually wears a normal sweater made by his grandmother and brown trousers.

## Jerry



**Age**: 10

**Jump distance**: Short

**Favorite candy**: Blue

One of the best friends of the main character. He always wears blue, to match the color of his favorite candy. Among his pastimes, he likes to draw, play videogames, and eating, which makes him not really sports-suitable. Since he’s a little chubby, he can’t jump very far to catch flying candies.

## Manfred

**Age**: 8

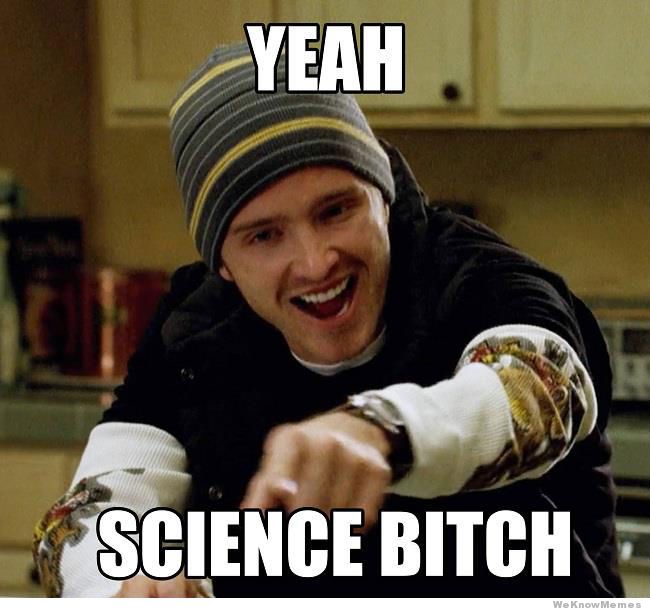
**Jump distance**: High

**Favorite candy**: Red

**Special**: Baseball bat

Manfred, contrary to Jerry, is totally nuts about sports. He loves candy, for sure, but his goals are to become a professional baseball player. Jerry can jump higher than anyone to catch flying candies with his baseball glove. Should a candy or a stone be addressed to hit him on his body, he will send them flying away with his baseball bat. He wears his white and red baseball team’s uniform.

## Jethro (Jet)



**Age**: 13

**Jump distance**: None (always flying)

**Favorite candy**: Purple

**Special**: Has a Jetpack, and is constantly flying up and down.

Over candy, there is one thing that Jet loves even more: Science. He has developed his own jetpack and only himself knows his own limitations on his inventions. Below his scientist coat, he wears purple.

## 

## Tony



**Age**: 9

**Jump distance**: Low (and flips his skateboard)

**Favorite candy**: Yellow

**Special**: He rides a skateboard, and doesn’t stop going left and right on the platform he is.

Tony is a hyperactive guy. He always has to be moving. Since he discovered skateboarding he always likes to feel the wind on his face while riding his board. He always wears a yellow T-shirt with a dark yellow helmet and ragged jeans. His dream is to become the best skater in the world.

## Hideo



**Age**: 7

**Jump distance**: High (but only for his favorite candy)

**Favorite candy**: Green

**Special**: He is camouflaged with the background and is hard to see. (He holds a rag to mimic the background)

Hideo was raised on a very traditional family of Ninjas. For him, camouflaging with the environment is as easy as breathing. His hard training has also given him the ability to jump very high when he sees one of his favorite mint candies.

# Types of Candy

Each candy behaves in a different way to the physics in the game. All of which are explained right here.

## Red Candy



**Gravity Reaction:** Normal

**Behavior against wind:** Normal

**Weight:** Normal

**Special:** Nothing

A standard strawberry-cream flavored candy. It’s easily the most popular and common of all candyfolk.

## Blue Candy



**Gravity Reaction:** Normal

**Behavior against wind:** Unaffected

**Weight:** Normal

**Special:** Bounces whenever it hits a wall or the ground.

This candy is partially made with jelly, which makes it equally soft and delicious.

## Green Candy



**Gravity Reaction:** Low

**Behavior against wind:** Very affected

**Weight:** Light

**Special:** Nothing

Mint flavored. It is particularly small and light, due to its intense nature.

## Yellow Candy



**Gravity Reaction:** High

**Behavior against wind:** Unaffected

**Weight:** Heavy

**Special:** Upon reaching the apex of the Parabola, it will plummet down in a straight line.

Lemon candy filled with gooey liquid. It makes it particularly heavy. Some people recommend this candy to be eaten, since it seems to go very well against throat pains.

## Purple Candy



**Gravity Reaction:** None

**Behavior against wind:** Normal

**Weight:** Normal

**Special:** This candy seems to ignore gravity entirely.

Nobody knows the mysteries of what these candies are made of. The greatest scientists seem to have figured out that the white stripes are made of cream. As for the purple part…

## Bubble gum



**Gravity Reaction:** Normal

**Behavior against wind:** Normal

**Weight:** Normal

**Special:** Upon hitting a wall or platform, it becomes stuck.

Standard tropical fruits bubblegum. Very Sticky. Be careful not to step on one.

## Stone



**Gravity Reaction:** Normal

**Behavior against wind:** Unaffected

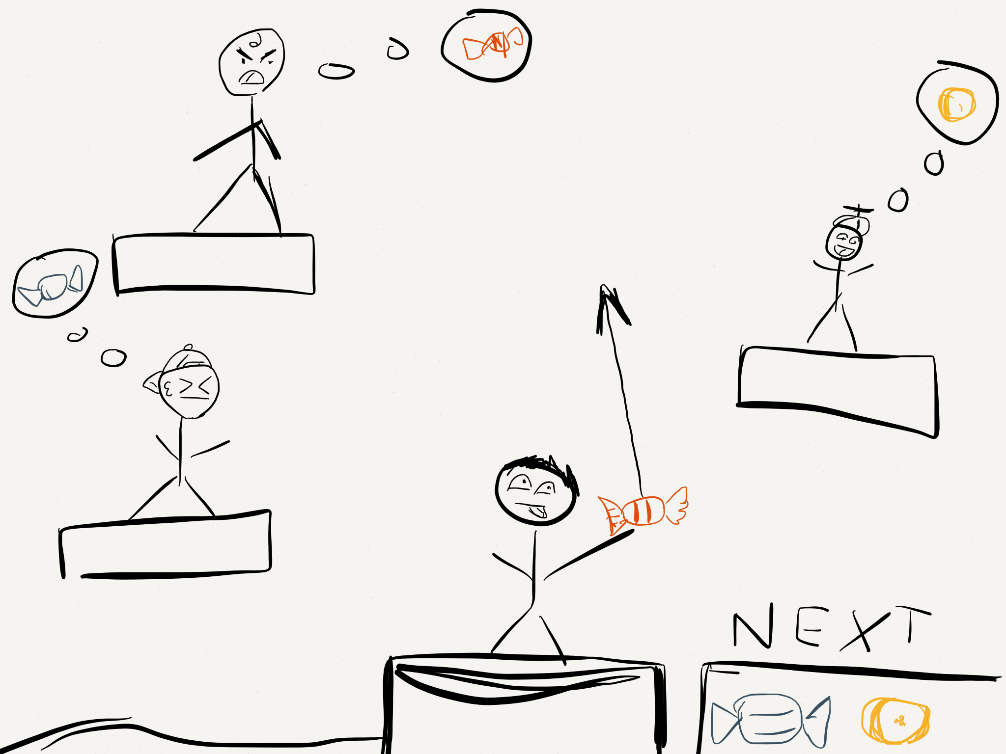
**Weight:** Heavy

**Special:** It can break glass, obstacles or item containers.

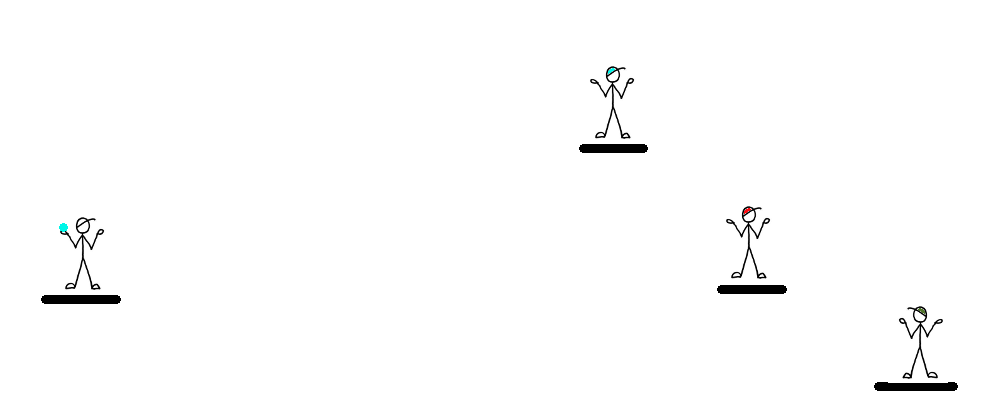
A stone. You can find them anywhere. Really, it’s just a stone. It works wonders when you use it to hold paper not to be blown by the wind. Do not eat. It’s an actual stone, not candy.

# Game Mechanics: Static and Scrolling levels

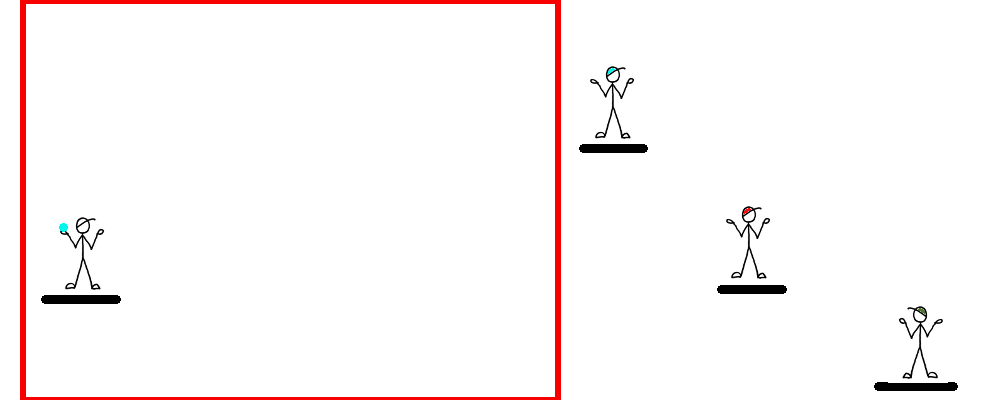
There will be two kinds of levels on the game. Static, and scrolling. Ideally on the first levels all the objectives will be displayed on the same screen, and there will be no need for scrolling. The levels will remain a fixed screen where all the player has to do is to aim and to shoot. An example of this kind of levels will be the one used before as an example of the game concept:



However, further in the game, the player might encounter longer levels that the visual area comprehends. For example, imagine this is the overview of one of the advanced levels:



The main character is located on the left, while the three kids are located on the far right end of the screen. If we wanted to make the whole level visible, it might render the characters way too small and gameplay would become inaccurate, that’s why we will keep always the same aspect ratio and show the exact same area, as explained on the next screenshot with a red rectangle.



The red Rectangle here shows what the user will be seeing. Bear in mind, though, that the screen can be scrolled at any time **only horizontally.** The game will not feature vertical scrolling at all. The player will have a visual help to know where the kids are located if they are not seen on-screen. Imagine this as “what the user is seeing”:



This (alongside with the other UI elements) is what the user will be seeing. How the screen scrolls, depends on the control method that we choose upon playing. If the player is using the **keyboard mode**, pressing the “left” and “right” arrows (since they are not used) will scroll the map horizontally. Pressing “up” and “down” are used for aiming.

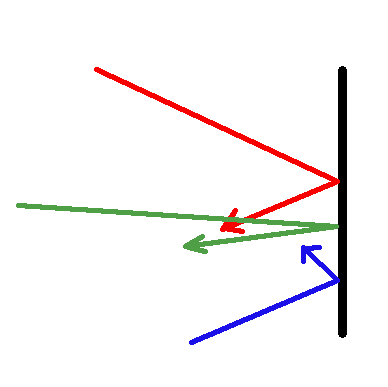
If the player is using **mouse,** the mouse wheel scrolls the map horizontally. Bear in mind, the player will not be able to shoot candies unless the user clicks on the main character.

# Game Mechanics: Gravity and Wind

Gravity and wind come defined by the Physx engine, so there’s not much to indulge around here. The page will remain open, in case we need to make specifications to it.

# Game Mechanics: Throwing and Airtime

Depending on the control scheme used, the throwing mechanic works slightly different, as explained before on the control methods. Now for what throwing implies…

* While the player is charging the throw, (or aiming if he’s using the Mouse controls) the main character will change to his “Aiming” sprite.
* Upon releasing the key, or the mouse, the main character will play his “Throwing” animation, comprehended by two or three frames.
* At this point, the object he throws leaves the main character’s hand with the potency and the direction given.
* The object thrown rotates its graphic as fast as determined by the strength of the throw. As the object loses its momentum, the rotating will slow down as well, until reaching the point where it doesn’t rotate at all if it’s falling to the ground.
* If the object collides with a wall or obstacle, depending on the nature of the aforementioned it will behave in one way or another.
  + For walls, standard behavior is that the candy (or stone) bounces off on the opposite direction to the one the candy was going, with 50% of the momentum it had upon colliding with it. Gravity laws still apply! Here’s a few visual Examples:  
    
  + There are two exceptions for these rule: The blue candy will bounce back with 100% of its momentum instead of 50%.
  + When bouncing on the ground, instead of bouncing back on the opposite direction, it will keep its direction.

# Game Mechanics: Materials and containers

Aside from the bouncing mechanics, there are certain factors that can influence on the flying object’s purpose, as well as the elements that might be included on levels:

* **Concrete:**
  + Mostly used on ground and walls
  + Standard bouncing mechanic (as described on the previous topic)
  + Unbreakable
* **Wood:**
  + Can be used for obstacles, or ground (platforms)
  + Standard bouncing mechanic
  + Breakable, but only if hit by **two stones.**
    - The first stone will visually **crack** the wood
    - The second stone will break it and it will disappear.
* **Glass:**
  + Can be used for obstacles only
  + Standard bouncing mechanic for candy only. Stones will go through it.
  + Breakable by a single stone with enough momentum, and will make it go through it.
  + It can happen that the glass is not broken by the stone, but it will have to go VERY slowly and hardly have any momentum at all not to break it.
* **Sand:**
  + Can be used only for ground. It can be placed on platforms, but only if there is another material to contain it.
  + Fluid-like properties (If the container is broken, it will fall as if it was water)
  + Nothing can bounce on it. If anything hits sand, it will stop moving entirely.
  + Unbreakable (obviously)
  + If mixed with **water**, it will turn into **mud.** The only property that will change on this is that nothing that is over it can move (or jump) at all
* **Water:**
  + Can be used for ground, or placed in containers like sand. It can also be placed on Piñatas or other containers, too.
  + Fluid-like properties (obviously)
  + Turns sand into mud
  + Things will not bounce on water, but sink.

Also, the game will have some containers that can help the player to solve the puzzles, or serve as obstacles. All of these containers will open upon being hit by a stone or a candy with enough strength (just like glass, but also with candies).

* Piñata: Can contain candies, stone, or water (or all of them at once). Upon hovering the mouse over it, the player can see the contents of the piñata on a bubble displaying it, like shown on the example:   
    
  If the player is using a keyboard, pressing the “Shift” key will show the contents of all containers on-screen while the button is held.
* Vases: These contain only water, and are used more as an obstacle than anything else. They can be only broken with stones, as if they were made of glass.