Platitude: Design Document

CS121 - Software Development | Fall 2010

Harvey Mudd College

Project team: Philip Aelion-Moss, Christopher Beavers, David Cook, Sarah Ferraro, Aaron Gable, Leif Gaebler, Jacob Heller, My Ho, Sean Laguna, Kathryn Lingel, Camille Marvin, Stephanos Matsumoto, Beatrice Metitiri, Emily Myers-Stanhope, Oliver Ortlieb, Thea Osinski, Richard Porczak, Kevin Riley, Max Strater, Rahul Swaminathan, Russel Transue, Heather Williams, Lilian de Greef, Alexandre François (pm).

Table of contents

Overview

Platitude is a hybrid that combines a platformer game with aspects of tower defense games.

Two types of players work individually or in teams to prevent the other type from winning various levels. One type, the platformer, tries to traverse a level in a set amount of time. The other type, a builder, uses money to create obstacles to prevent the platformer from getting to the end of the level. The platformer gains abilities and becomes stronger as the level progresses while the builder gets access to new obstacles/abilities and acquires more resources. The game is playable as a single player (either platformer or builder) and with multiple players arranged in teams that work together.

Game Mechanics

Platitude combines a platformer game with aspects of tower defense games.

Two types of players work individually or in teams to prevent the other type from winning various levels. One type, the platformer, tries to traverse a level in a set amount of time. The other type, a builder, uses money to create obstacles to prevent the platformer from getting to the end of the level. The game is playable as a single player (either platformer or builder) against AI and with multiple players arranged in teams that work together.

In every game there is only one builder, but there can be up to 5 platformers working together. If there are 2 human platformers, then there will be 3 AI platformers.

Both types of players start the game at the same time, but, to give the builder some time to prepare, there is a predetermined portion of the map that the platformer must traverse before it gets to the portion of the map in which the builder can build. Each level has a set amount of time during which the platformer must traverse the entire level. If the platformers fail to do this because they run out of time or lose all their lives, the builder wins.

Every player has a profile that they login to. A user's profile will keep track of the number of games played and their win/loss record. It will also break down stats by player type so each user can see how well they do as a builder and as a platformer.

Overview of a Game

General Play

The player starts the game. If they wish to play with/against other real people, they connect to a central server on which the game will be hosted. If they wish to play alone, they do not need to connect to a server. They then choose to join an established game or start a new game. If they start a new game, they can choose to be either a Platformer or a Builder. If they join an established game, and there is already a Builder, they must be a Platformer.

When the creator of a game decides that they have enough players (including 0 other players, if playing alone) then they can decide to start the game. Any character slots not filled by players will be filled by AIs.

The players then play the game. The game is divided into levels. A level ends when either the Platformers or the Builder defeats the other team. A player can choose to leave the game at any time. If they do, their character is replaced by an AI and the other players are notified of the change. The same happens if a player's connection drops for more than 15 seconds. A player can choose to join a game in progress as well - they will be given the option to replace an AI when the current level ends.

When a player leaves, their statistics for that game are displayed. When the last player in a game leaves, the game is ended and overall statistics are displayed. If either the Platformers or the Builder win the game overall (not just winning a level), then individual and overall statistics are displayed to each player. The game is then closed, and players are allowed to join or create a new game again.

Platformer Play

The Platformers are first given a view of the whole map (terrain, not obstacles) and a chance to strategize while the Builder begins placing obstacles. Then, after a certain period of time, the Platformers spawn at the beginning of the level and begin to work their way towards the goal. Along the way, the Platformers use their abilities - including jumping, climbing, and using items and powerups - to get past the obstacles placed by the Builder. The Platformers have a limited number of lives, and every time they die they respawn at the most recent checkpoint (which is an item placed by fellow Platformers) with one fewer lives. When a Platformer reaches 0 lives, they are given a Builder-style zoomable view of the whole map so they can observe the rest of the level. If the other players so choose, they can donate their lives to a dead player to allow them to respawn. In a game with only one player, that player could "steal" lives from the AI characters in order to respawn. The platformers can pick up items that they can then choose to use whenever they want. The effects of power ups are felt immediately.

If the Platformers make it to the end of the level with lives remaining, they win that level. There may be a boss-type obstacle for the Platformers to defeat near the end of the level.

Builder Play

After entering a game as the Builder, the builder is presented with the starting map and a set of resources. The Builder can use those resources to create/buy obstacles for the Platformers. These obstacles include towers, terrain-changing tools, and monsters. The Builder can place the obstacles in areas of the level that are designated as "buildable". The Builder can zoom in and out of the map arbitrarily to get a better view of certain areas to decide where to place obstacles.

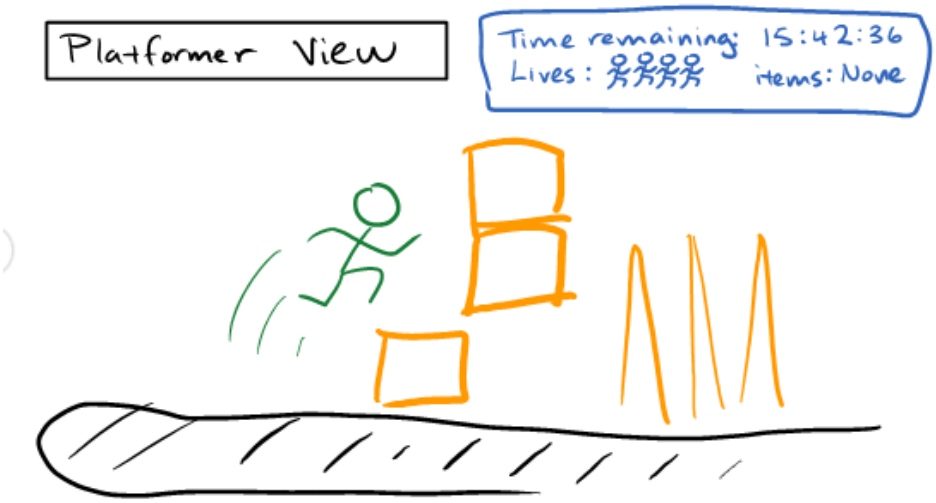
The Builder has a period of time at the beginning of the game to place obstacles before the Platformers enter play. Once the Platformers begin to progress through the level, the Builder can continue to place obstacles. The Builder gains additional resources every time an obstacle kills one of the Platformer characters. The Builder receives an income every x seconds. This income increases as the Platformers progress through the level.

If the Builder manages to remove all lives from all Platformers before they reach the end of the level, or manages to hold off all of the Platformers for a designated period of time, the Builder wins that level.

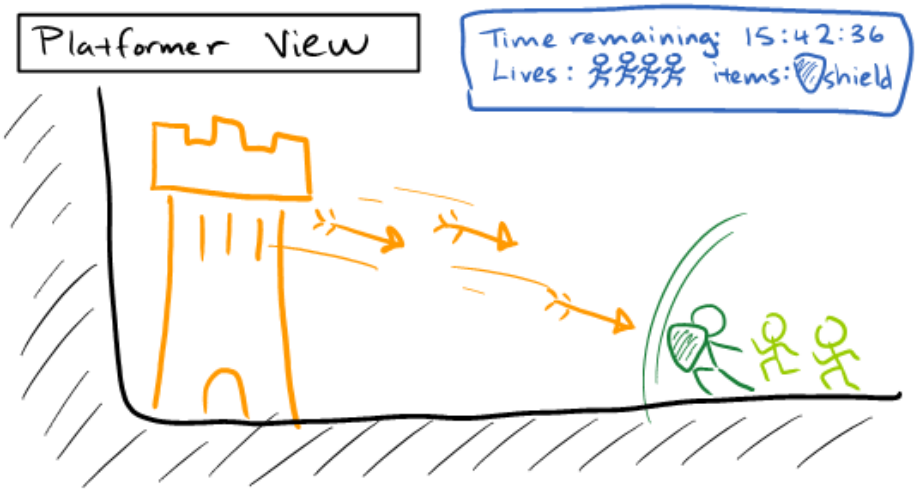
Platformer Mechanics

The platformer has a limited number of abilities (jump, move left, move right, etc.) which it can use to traverse the map/level. However, it can gain more abilities as it progresses through the map. The map should be completely traversable by the platformer before the builder starts creating obstacles. The platformer cannot see the entire map, only a small portion of its immediate surroundings. In addition the platformer has a set amount of health. Any damage he takes is subtracted from his health, and when he reaches zero health, he must start from the beginning or most recent checkpoint.

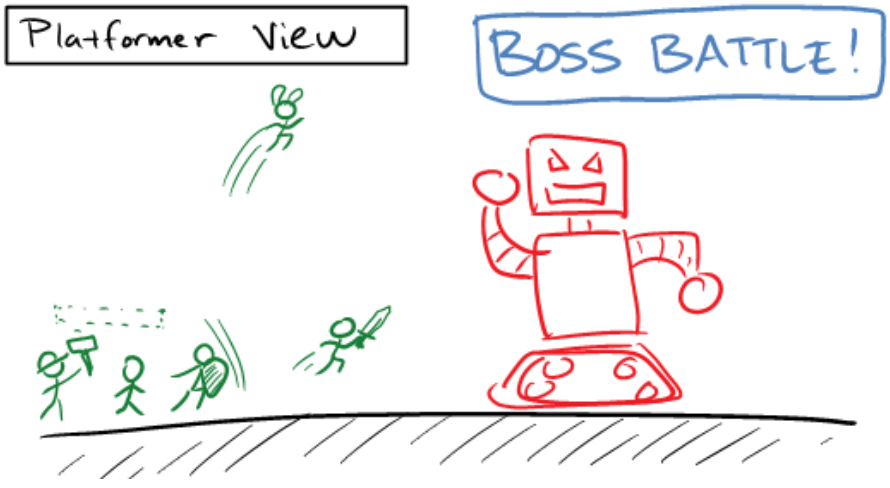
The adventurers/platformers can play the game as a regular platform game, trying to get to the end within a limited amount of time and number of lives. When they die, a life is subtracted from that player's total. When they run out of lives, they either enter an observer mode, or can be given more lives to continue playing by their teammates.



Players can acquire items or abilities to make the game play more interesting. A possible example is a shield, shown below. It would be really cool if items allow players to work together to reach their goal. The number of items a platformer can carry in his/her inventory and equip as power-ups is limited (exact inventory size will be defined during testing, let's limit players to one power-up for now).



A possible ending to a level could be a boss battle, as is typical in a platformer game. (Would the boss be controlled by the builder? A possible way to implement the boss would be to allow the builder to spend resources on abilities for the boss, such as fireballs of various strength, jumping into the air and landing on platformers, etc. They would have to decide how to distribute resources to the level and the boss in the most effective way. More resources used in the level could mean the platformers never reach the boss, more resources used on the boss could mean the platformers can't beat it. It would be more in the spirit of a tower defense game if the builder can't directly control the boss, but they just power it up and let it go.)



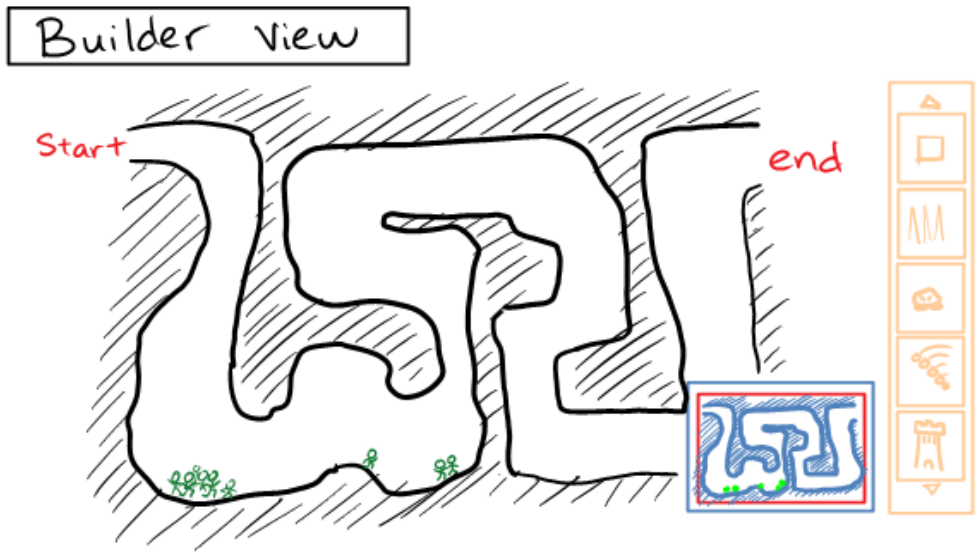
Platformer Abilities:

- Movement
  - Move (walk/run) left and right
  - Jump (between 3 to 5 times platformer height)
  - Climb (i.e. for cliffs, ladders, etc)
  - Crouch
- Use items
  - Pick up, carry, and place/throw one-use items
  - Equip power-up items (automatically happens upon picking it up)
- Interact with some obstacles
  - eg. pull lever, push button, etc

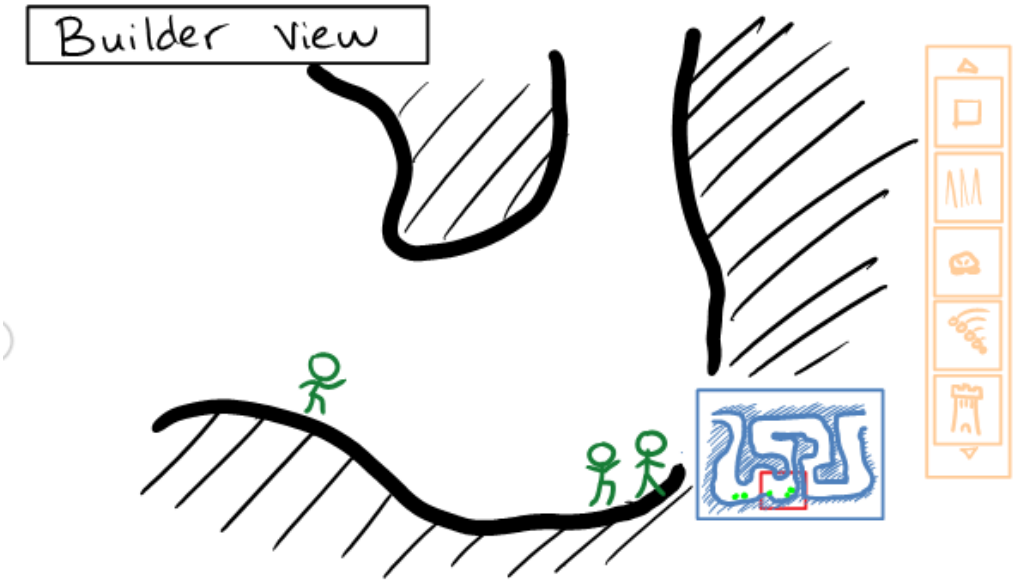
Builder mechanics

The builder doesn't control a specific character on the screen, instead it uses resources to buy objects and strategically place them in the map to prevent the platformer from getting to the end of the level.

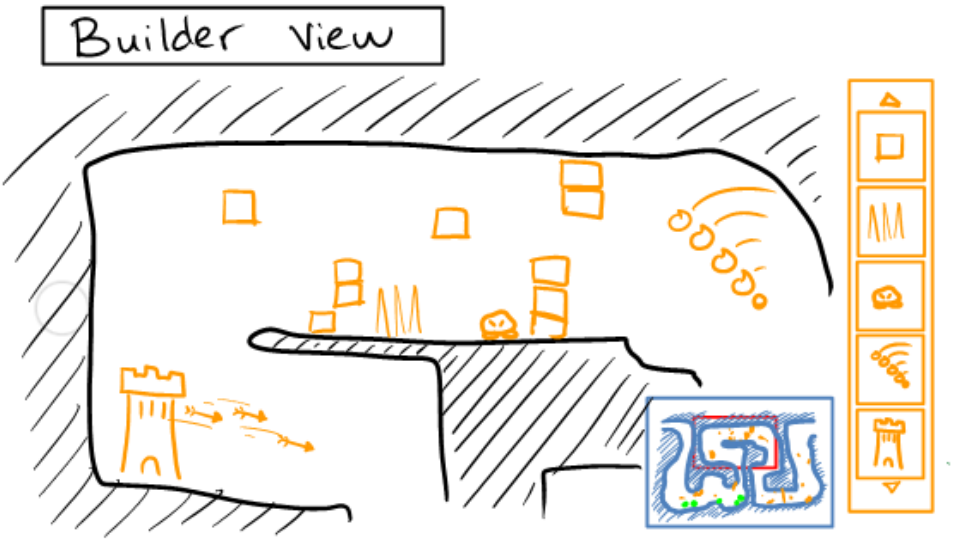
The builder can see the whole map including where the platformer is. A possible depiction is shown:



The Builder would also be able to zoom in wherever and however much (s)he pleases on the map, seeing everything in real-time.



The Builder can build obstacles to make traversing the path more challenging, with the intent of slowing down the adventurers / platformers. A possibility to illustrate the concept is show below:



Builder Abilities:

- View map
  - zoom in/out of areas
  - pan through areas
- Place obstacles
- Upgrade obstacles
- Gain resources for acquiring obstacles
- Gain/unlock/buy obstacles

Multiplayer Mechanics

This is also discussed in "Before starting the game," below, and Game mechanics above

Someone creates a game in the "player lobby" and either invites who else can play (possibly no one, if he/she wants to play solo) or opens the game to anyone in the lobby. Other people join the game accordingly. The game map is chosen upon creation.

Upon starting/joining a game, a player selects which role he/she plays. The screen displays the decisions of each player. The game begins when everyone agrees they are ready (eg. they click "start") and they start playing.

If a player quits or loses connection, their corresponding part / character in the game likewise does nothing for a preset duration (eg. 30 seconds). If he/she does not return to the game or regain connection within that time-frame, an AI takes over the role.

Between levels, players can choose to leave or join the game (which would be especially helpful with connection problems, in case someone wants to get back into the game and the game isn't capable of letting someone just jump in, or if someone shows up late).

Players that leave the game voluntarily (not a connection problem - they specifically indicate that they quit), then a loss will be added to their record. If players lose when they quit, they have an incentive to play the entire game.

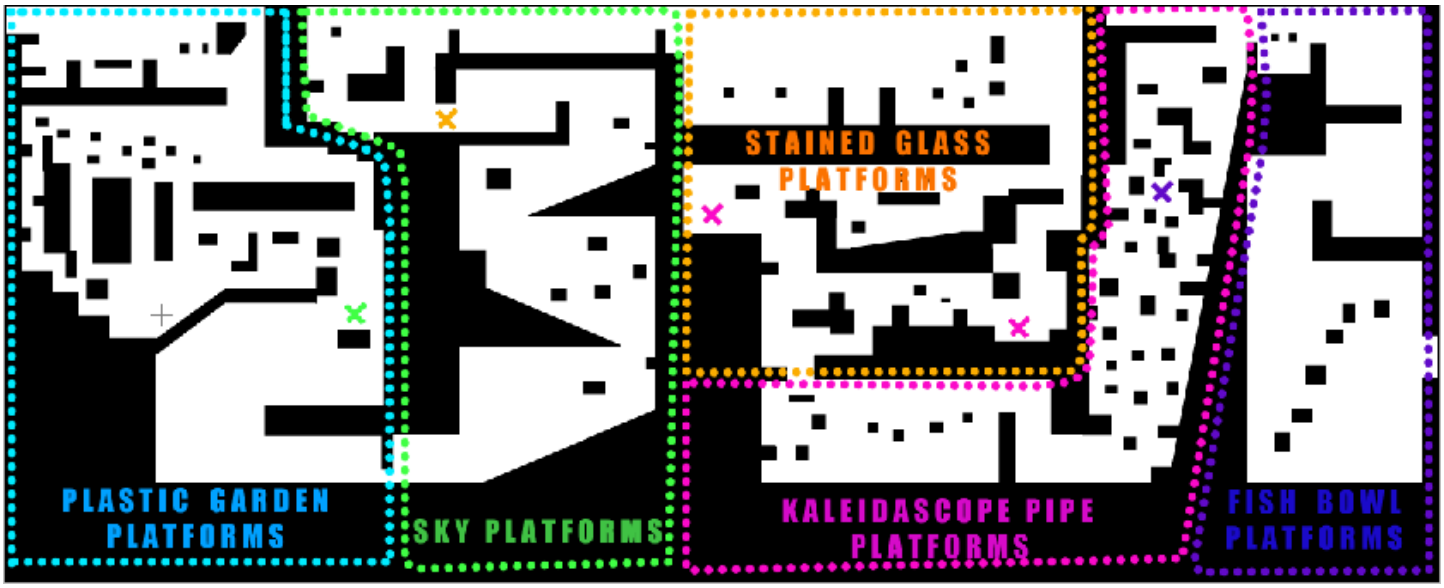
Game elements

Level map

A level map is a rectangular area of arbitrary size (for reference, use the platformer height as the unit), which comprises of:

- A background image (background graphics do not interact with game elements, just there to be pretty)
- Regions of interest:
  - Platformer entry
  - Platformer goal
  - Climbable
  - Buildable
- Platforms (cannot be blown up)

Concept Level Map:



The concept here is that there are five distinct portions of the map. Each with it's own theme. Prebuilt platforms and builder platforms are distinctive in each of the five portions and themed as listed above. When the platformers pass each of the "checkpoints" (marked as X's) the builder gets an increased income so that he can afford new platforms and obstacles for the next colored portion.

This is a mockup of the level map, and will be filled in with the proper theme as the game progresses. (i.e. it won't be black polygons forever).

Link to Full-Sized Level Map: http://i53.tinypic.com/e04fug.png

Objects

Platformer Items

Items the platformer uses:

- Pick-up and place/throw items. These go into a player's limited inventory when picked up.
  - ladder: used to climb
  - movable platform: a special kind of platform players and pick up and place where they please
  - grenade-bomb: bombs that obliterate things within a certain radius upon contact when it's thrown
  - time-bomb: obliterate anything within a certain radius of bomb, x number of seconds after it gets placed
  - checkpoint: set by platformer; takes time, the platformer must stand still; when the platformer dies, it will respawn at the nearest checkpoint
  - portal: transports the platformer through time and/or space; time portals will move the platformer back in time without taking time off the game clock; if a platformer uses a 15 second portal 1 minute and 17 seconds into the game, then the level will go back in time to what it looked like at 1 minute and 2 seconds into the game, but the game clock will still say 1:17.

- Absorbable, one-use items. Players absorb these upon contact. They take effect immediately.
  - extra lives: whenever the platformer comes across and extra life, the life is added to their total; it is not consumable at any time like the other abilities
  - health recovery: used at a time designated by the platformer; returns a specific amount of hitpoints to the platformer; there can be different strengths of healthpacks

- Power-up items. Players gain special abilities immediately upon picking up the item. A player can only have one such ability at a time. A player will drop their former power-up upon picking up another, leaving old one available to anyone else.
  - shield: blocks a set number of hitpoints; there can be different shields of different strengths
  - high jump: makes the platformer jump higher
  - speed boost: an ability that can be used at any time; makes the platformer move faster (maybe make this an absorbable item? -LD)
  - silk shooting: ability to shoot out sticky rope to trapeze from one place to another, spider-man style

Builder items

Items Builder can place:

- Terrain modifications
  - blocks
  - platforms

    - walls
    - pit: platformers die if they fall down a pit
    - "assassin" item: the builder gets to go into the stage and mess with the platformers, hand-to-hand
        - we talked about this a while ago...still a possibility?

- Platformer hazards
    - spikes: die/significant damange if land on spikes, but running into spikes horizontally does not result in their death. these have a limited size (width) that can be increased by placing multiple spikes/pits close together
    - small monsters: they walk about and harm platformers if they collide
    - towers: towers must be places on horizontal ground; there are different kinds of towers that shoot different kinds of projectiles (simple arrows/bullets, slowing shots that reduce the movement speed of the platformer, etc.); towers can also spawn monsters instead of shooting things; towers that shoot and monsters must have AI in order to target and attack platformers automatically (upgrade increases the effectiveness of the tower)

- Sections with special attributes (the builder charged by surface area)
    - sections with "special goo": the goo coats a surface and causes the platformer to speed up or slow down depending on the type of goo
    - inverted gravity
    - Fans/jump pads - basically a slight impulse force along a given direction that may accelerate/slow down P in order to make a given section more complicated

- Teamwork traps (only available when there are multiple platformer players; requires platformer cooperation in order to pass)
    - two levers must be pulled at the same time in order to open a gate

Characters

Mostly monsters that can either remain stationary or patrol an area.

Graphics

Graphics are 2-D, vectorial, layered. Different roles have different views.

Platformer

The platformer view is that of your average side-scroller. It consists of the graphics for the player, the background level, the obstacles placed by the builder, and icons indicating health, items, and time remaining. There will also be an icon at the bottom right of the screen that allows for simple control over music, like volume and mute.

The screen size / dimensions (tentative): let the height of the platformer avatar be 1 unit. The platformer's screen size is then 16 x 9 units.

The view presented to the platformer is fixed and scrolls automatically to keep the character within a central window.

Builder

The builder view allows the player to view the level at various depths of zoom. Because of this, the graphics requires the integration of a zoom feature into the builder's menu.

The builder's view comprises two distinct sections: the level view (featuring zoom) and a menu for objects to drop. It may also be useful to relay graphical information regarding the state of the platformer. The build menu must incorporate remaining build resources as well as time remaining. It would help to show acceptable/unacceptable locations for placement of the object that the builder is trying to place. For example, if the builder was trying to place a tower, any non-horizontal terrain would be highlighted in red and horizontal terrain would be highlighted in green.

User Interface

Spectator Input

The basic controls for the spectator client are as follows:

- a: move left
- d: move right
- s: move down
- w: move up
- '=' and '+': zoom in
- '-' and '_': zoom out
- 'g': view whole map
- 'f': full screen
- 'F': undo full screen
- click-and-drag mouse input: move view of map

Spectator Output

The spectator can only view the world, not interact with it, so all relevant input will result in a change in the view of the map.

Platformer Input

The basic controls for platformer motion are as follows:

- a: move left
- d: move right
- s: crouch, climb down (if on a climbable object)
- w: climb up (if on a climbable object)
- [space]: jump

The platformer also has the ability to interact with items and objects in the level. Those interactions are controlled as follows:

- pick up item: e (while adjacent to the item), or right-click on the item (when close enough)
- use current item: r, or left-click on the item in the inventory (if we allow multiple items in the inventory, then 'r' will be replaced by the number key corresponding to the item in inventory)
- open/close inventory: i, or left-click on the inventory icon on the screen

Platformer Output

When the platformer presses the movement keys, it will execute the resulting movement on screen. If the jump button (or any of the other innate movement buttons) is held down, the platformer will jump (or move appropriately) until the button is released. Adding items to the platformer's inventory will make them disappear from the map and appear in the platformer's inventory. If the inventory is full, then trying to add something will result in an error message on the screen. Picking up and using items will all have sound effects. Each item should have a different sound effect when it is used, but they can all have the same sound effect for when they're added to the platformer's inventory. Opening the inventory causes a panel to appear on the side of the screen, displaying the current item(s) in the inventory, and partially obscuring the platformer's screen.

Builder Input

The builder will be able to manipulate their view of the map as follows:

- a: move left
- d: move right
- s: move down
- w: move up
- '=' and '+': zoom in
- '-' and '_': zoom out
- 'g': view whole map
- 'f': full screen
- 'F': undo full screen
- click-and-drag mouse input: move view of map
- click in mini-map: jump main view to center on selected point

The builder will have a construction pane showing the building and upgrading options. They can interact with this pane in two ways: by clicking buttons on it, or by pressing numeric hotkeys that are associated with the buttons in logical order. Clicking a button or using the hotkey corresponding to an object that can be placed on the map will bind that object to the mouse. The builder can then click while the object is over a buildable area in order to place the object there.

Builder Output

The screen will immediately respond to the builder's resizings and movements. If the view is already zoomed out to 100 the screen will not respond to further attempts to zoom in that direction. When an object is selected, an overlay of a grid will appear on top of the map that will show the builder all of the places where she can put the object. Once the object is placed, the overlay disappears, and the object is now part of the map. The computer will display an error message and play an appropriate noise if the builder tries to select on an object that is too expensive (or unavailable).

Music

Definition

The music engine is essentially a background track generator that uses the builder's placed objects as loops.

How It Works

The map is minimalistic at the beginning of the level. The background track is accordingly also minimalistic, with only a very basic beat and possibly a basic loop. This basic beat might suggest a heartbeat - when only the platformers are on the stage, it might be cute to suggest that they're nervous and their hearts are thumpin' in anticipation of the level ahead.

The background track gets more complex as the builder adds more objects to the map, and the tracks also reflect the objects present in the map. So, while the addition of a single object might not change the tracks being played, it will factor into the overall scheme for which tracks are played.

Tracks might be tied to:

- a type of object (probably an important type of object) being displayed on the screen
- a combination of types of objects
- a certain amount of a type of object
- player health (when a player is about to die, all the tracks are cut except a "heartbeat" track...we'll flesh this concept out later)

Object location does not affect characteristics of the loops (although this could be changed to do so), and so listening carefully to the background track only reveals what it in the map, and not where those objects are. The tempo of the background track is adjusted according to how close the platformer is to the end of the level. As the platformer nears the end of the level, the tempo increases. (This could also be changed by other parameters, such as time remaining or health.)

Basically, we want the song to sound good regardless of what's on the stage. We also don't want the tracks to be switching on a second-to-second basis, which will just sound muddy and confusing. Also, we don't want the sound to directly reflect the items on the stage, but suggest in a broader way the STATUS of the stage. Busy songs mean the platformers have a lot of work to do, and chiller songs means smooth sailing.

Physics Simulation

The game contains a 2D physics engine that implements gravity, friction, and collision detection. This engine allows a platformer character to walk on solid ground, jump, and climb. Walking off a platform will cause the platformer to fall according to gravity. Steep inclines will cause the character to slide. Other objects (obstacles put down by the tower defense side or platformer powerups) will also fall with gravity and slide on steep inclines. The physics engine will implement a velocity vector, an acceleration vector, and mass for each object affected by physics.

AI

Game should be playable by one person regardless of which type they are, so there must be AI platformers and builders.

Towers

Simple state machines will do

Monsters

Simple state machines as well.

Bosses will be controlled by the builder.

AI Platformer

An AI platformer will try to get to the end of the level as normal (human) player would. It should know when to set checkpoints and when to use healthpacks etc. It shouldn't be able to make a perfect jump every time.

AI Builder

An AI builder will combine the different objects in various ways to make the platformer struggle to find the best way to pass certain areas. It should be able to use the changing terrain of the map to its advantage.

Ideas on how to do this: What about making a simple scoring system for how well certain items are placed - near cliffs would give better scores because those areas tend to be more inherently dangerous, etc. Then, generate a bunch of random placements for items throughout the map, and take the highest scoring ones...and then make the builder place them on the map in some timely fashion? And when more resources become available, just rinse and repeat?

Game Progression

There aren’t many levels in the game (maybe less than 10), but because platformers will face different builders and builders have a lot of freedom to create different obstacles on every level, levels will rarely look the same. The objective is more to beat your opponent than to beat a specific level/map.

Platformer Objectives and Progression

The platformer tries to get to the end of the level as quickly as possible without losing health or lives. If the platformer completes the level in the allotted time, they win. This win goes on their record, which other players can see.

Builder Objectives and Progression

The builder uses resources to place objects in the platformer's way. The builder's income increases as the platformer

progresses through the map. If the builder prevents the platformer(s) from reaching the end of the level in the allotted time, the builder wins. This win goes on their record, which other players can see.

income and obstacle availability:

- begins with base-line income of resources
- extra income rate proportional to progress of platformers (i.e. if the average progress of all platformers is 42%, the builder's income is 42% higher than the base-line's)
- if platformers pass certain checkpoints, new obstacles are available for purchase given enough resources
- in the 60 seconds, the builder got more money because the platformers lost lives...do we want to maintain this functionality?

Question: What do they win? a medal? experience? access to better stuff?

Demo Level

The demo level is a simple set of platforms for the platformer to navigate while the builder adds to the level by placing objects.

Narratives

Sixty seconds of Play

Platformer (P) Builder (B)

Players load a map, and after a three second countdown, the game begins: the platformer starts trying to get to the end of the map, and the builder starts placing obstacles in the platformer's way. The platformer moves as quickly as possible while the builder creates obstacles quickly and strategically. The builder places a tower that shoots arrows at the platformer when it is in range. When the platformer gets to the tower, it pops a shield to protect it. After the tower, the builder constructs a bunch of platforms with spikes underneath. The platformer sets up a checkpoint after the tower so that it doesn't have to go past it if it dies. The platformer tries to jump from platform to platform, but misses and falls on the spikes losing a life. This causes the platformer to respawn at the checkpoint that it set up, and it immediately starts trying to jump from platform to platform again. With the new resources from killing the platformer, the builder decides to buy a monster to patrol a later portion of the map.

Sample first sixty seconds of play as Platformer

The game starts, the level map loads.

A countdown reaches 0 and player P takes control of a platformer character. The platformer is amidst a bunch of other platformer characters (a maximum of 5, be it AI-controled, human-controled, or both), colored differently to clarify which character P controls. All the characters start charging forward to reach the end of the map as quickly as possible.

P jumps over a ditch, hops onto a platform to avoid spikes, leaps onto other blocks/terrain in the way, and pauses to wait for a sweeping arm of fire to pass. With the right timing, P then runs forward again, closely at the heels of some characters and leading the way for others.

P doesn't time a jump quite right and accidentally collides with a small monster. As a result, P's health-bar drops.

P conveniently finds a cookie lying around. Upon touching it, P absorbs the cookie and regains health.

P also finds a ladder leaning against a block and picks it up, adding it to his/her inventory. The inventory can hold a limited number of items (perhaps a maximum of 5). Later, P finds a cliff that is a bit too high to jump onto. Thus, P whips out the ladder and places it against the cliff wall. Using the ladder, the platformer characters climb up.

A section of the map in front of P looks like it has a different aura (glowing another color, glittering, or something). P does not pay enough attention to his/her surroundings to notice and runs into it, and finds that gravity completely changes. P then "falls" upward and suddenly runs around on the ceiling, upside-down.

Because P is not used to the direction he/she jumps when upside-down, P fails to jump over another set of spikes, loses all of his/hear health, and dies. As a result, the platformers lose one of his/her lives. P immediately respawns. Fortunately, another platformer found a checkpoint-item and placed it a fair distance beyond the start of the map! Thus, P respawns there and does not need to traverse the previous sections of the map.

P finds a shield lying around on the ground, picks it up, and becomes equipped with it. P now has the ability to deflect some kinds of weak attacks!

P discovers that the path has changed a bit since he/she last traversed it - the builder has added a tower that shoots arrows! The shield lets P get close without careful dodging and also protects all the other platformers crouching behind. One of these platformers picked up a gernade-bomb earlier, so he/she/it throws the bomb while under P's protection and damages the tower. The arrows stop shooting out and all the players can run past safely without careful dodging.

First sixty seconds of play as Builder

The game starts.

The level map loads. A 3 second countdown precedes the start of play.

The builder has a starting income that she can immediately use, so she places some blocks and platforms with spike below them at the beginning of the map to make the platformers jump carefully across this section. While the platformers works on this, the builder constructs a tower that shoots arrows at the platformers and pours some slowing goo in front of it. After the tower, the builder places a patrolling monster to force the platformers to jump and avoid it. By this point, the platformers has gone through 25% of the map, so the builder's income has now increased. The next portion of the map is a vertical section with climbable walls and platforms that the platformers can jump on to get through the area. The builder pours slowing goo on the walls and goo that speeds up movement on the platforms to make it more difficult for the platformers to jump between the platforms. Now the builder doesn't build anything for a little in order to save up her income so that she can build a lot of stuff at the end of the map. Once she has enough saved up, she builds a tower and upgrades it a few times. Then she places blocks in front of it to make the platformers maneuver between the blocks while dodging the shots from the powerful tower. (end of first 60 seconds)

Before starting the game

The player logs into the system. They see their profile screen, where they are presented with their stats.

The player joins a player lobby where they can see other players and challenge people to a game, and have access to simple music options.

Someone can create a game in the player lobby and either invite other players (possibly no one, if he/she wants to play solo) or opens the game to anyone in the lobby. Other people join the game accordingly. The game map is chosen upon creation.

Upon starting/joining a game, a player selects which role he/she plays. The screen displays the decisions of each player. The game begins when everyone agrees they are ready (eg. they click "start") and they start playing.

This is also discussed in "Multiplayer Mechanics," above