MOBILE MULTIMEDIA AND GAMING

GROUP ASSIGNMENT

CT041-3-3

Members: ANG CHEE SIAH (TP038259)

LIM ZHI DA (TP041644)

SEE RONG JIE (TP038306)

Intake code: UC3F1805CGD

Module: MOBILE MULTIMEDIA AND GAMING (CT041-3-3)

Lecturer's Name: DR. CHEN TET KHUAN

MR. JACOB SOW TIAN YOU

Submission Date: 5th SEPTEMBER 2018

Table of Contents

Introduction

This assessment focuses on delivering a game in the format of mobile application, also known as the mobile game. In this case, we were given the theme of Puzzle, where the mobile game must include puzzle-solving elements of any kind, and have certain degree of continuity. The requirements of the output are as follows:

- ✓ Game must be playable in mobile platform (either iOS or Android platforms)
- ✓ Game must include puzzle elements
- ✓ Game must be able to run for at least 3-5 levels, if there is any need to implement levels.

In this case, the output of this assessment is decided to be a storytelling game with inclusion of a semi-roguelike puzzle, where the platform is expected to be in Android platform, made via the Unity Editor.

Literature Review

The review of the following games has been conducted among the members respectively:

Members	**ANG CHEE SIAH**	**LIM ZHI DA**	**SEE RONG JIE**
Game Reviewed 1	*1BeatHeart*	*Inside*	*Mekorama*
Game Reviewed 2	*Dungeon Cards*	*Samsara*	*Monster Strike*
Game Reviewed 3	*Potion Maker*	*Snipperclips: Cut It Out, Together*	*Causality*
Game Reviewed 4	*Dungeon & Girls: Card RPG*	*Lara Croft Go*	*Divine Gate*
Game Reviewed 5	*Baldi's Basics in Education and Learning*	*The Witness*	*Where Shadows Slumber (Demo)*

Game Reviews: ANG CHEE SIAH

1) *1BeatHeart*

- ✓ **PUBLISHER**: △○□× (Miwashiba)
  OTHER GAMES PUBLISHED: Alicemare, LiEat, 1BitHeart
- ✓ **GENRE**: Adventure-Platformer
- ✓ **DESCRIPTION**: A couple investigating strange murder cases that occurred in a resort hotel.

HOW TO PLAY:

Encounter a case, find evidences and testimonies, interrogate suspect & find contradictions within the suspect's lies by presenting collected evidences (vgperson, 2017)as in ***FIGURE 1.01***.

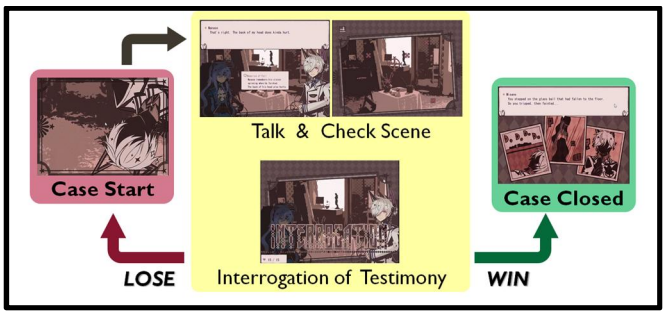


FIGURE 1.01: General Overview of 1BeatHeart Gameplay

In an example of the gameplay, the main character, Misane, is required to check the cause of her partner being unconscious, and begins the gameplay of a 'chapter'.

There are a few steps that occur during the gameplay, with reference to ***FIGURE 1.02***:

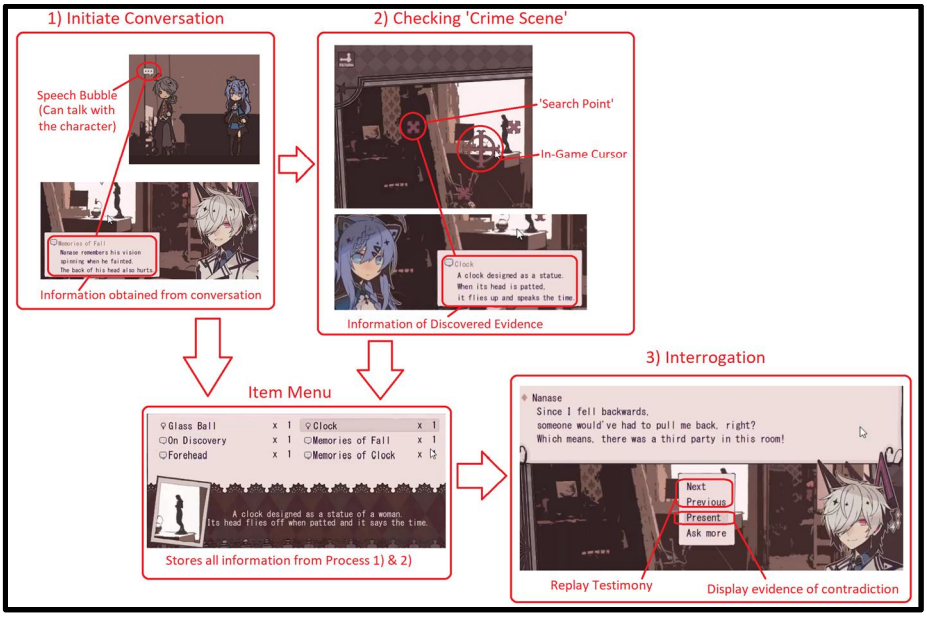


FIGURE 1.02: 1BeatHeart Gameplay Example (1 Chapter)

1) Initiate **conversation** with interactable characters (shown by floating speech bubble) to gather intel. (Information would be stored in Item Menu)
2) Check the 'crime scene' and collect **evidence** by searching through the area. (Evidence would be stored in Item Menu too)
3) Begin **interrogation** with the suspect and **find contradictions** in suspect's testimony by presenting the collected information.

- The player is given 15 chances, referred as '15 HP' for each chapter in the game, and each wrong option chosen in interrogation would result to 1 HP being deducted.
- The game is over when the HP is deducted to 0, and the current chapter would restart.
- While when the game is cleared when all contradictions in the suspect's testimony has been resolved, the game would advance to the next chapter until all chapters are cleared.

WIN CONDITION

- Success in interrogating culprit/Uncover all culprit's lies

LOSE CONDITION

- 'Heart'/ 'HP'/Chance drops to 0

GAME MECHANICS

❖ *CHARACTERS*

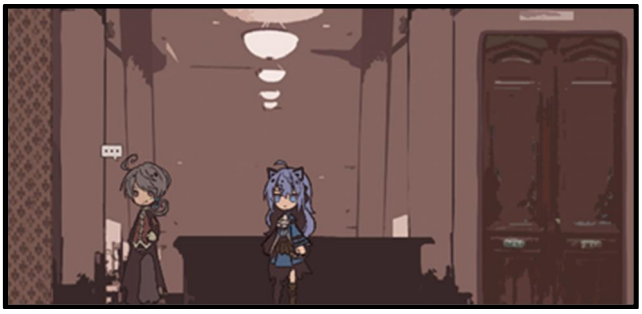
➢ Evidence providers OR Interrogate targets/culprits (shiba, 2017)

❖ *ITEMS*

➢ Evidence & certain Testimonies/Alibis are kept in Item Menu as items to trigger-use in Interrogations

❖ *ENVIRONMENT*

➢ 2D Platformer-based environment



2) Dungeon Cards

- ✓ **PUBLISHER**: 717 Pixels (Pixels, 2018)
  OTHER GAMES PUBLISHED: NONE
- ✓ **GENRE**: Roguelike-Puzzle
- ✓ **DESCRIPTION**: An adventurer exploring in a dungeon to earn money while destroying monsters and traps

HOW TO PLAY:

Move horizontally or vertically on a 3x3 dungeon card map and get as many money/gold as you can. Kill monsters and traps or get killed by them instead as seen in ***FIGURE 1.03***.

FIGURE 1.03: General Overview of Dungeon Cards Gameplay

In general, the player is equipped with Health Points (HP) where the moment the player touches the monsters or traps while moving, the **player's HP would be reduced** by the **value** the monster or trap carries as shown in ***FIGURE 1.04***. If the player is **equipped with a weapon**, the **weapon's HP** would be deducted instead of the player's.

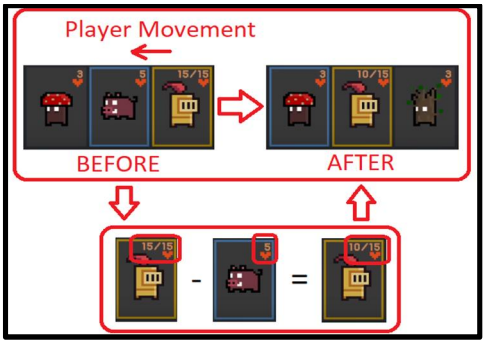


FIGURE 1.04: Damage Calculation in Dungeon Cards

During gameplay, the player could obtain **money** from destroying monsters or traps, where the accumulated money as shown in ***FIGURE 1.05*** could be used to **unlock new characters or powers** that brings benefits such as healing the player, damage immunity to certain traps, and so on.

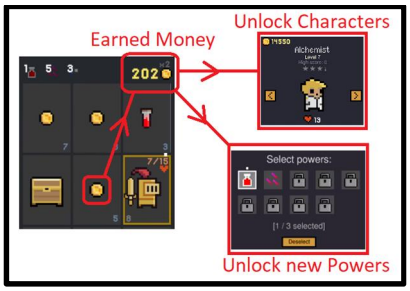


FIGURE 1.05: Money Usage in Dungeon Cards

The game ends when the player's HP hits 0. The amount of money obtained would count as the 'high score' of the game. Therefore, this game is of a endless run.

WIN CONDITION

- None

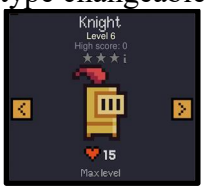
LOSE CONDITION

- When player HP drops to 0

GAME MECHANICS

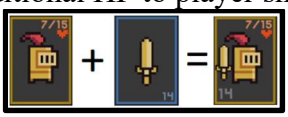
❖ *CHARACTERS*

➢ Player Card (Player Card type changeable at Menu)



❖ *ITEMS*

➢ Weapons (Provide additional HP to player shown in bottom-left)



➢ Money (in-game Currency)

➢ Monsters & Traps (deals damage based on displayed value upon contact)

➢ Potions (heals player by the displayed amount upon contact)

➢ Powers (triggered when Money is obtained to certain amount)

❖ *ENVIRONMENT*

➢ A 3x3 Card grid layout in 2D format

3) Potion Maker

- ✓ **PUBLISHER**: Sinsiroad
  OTHER GAMES PUBLISHED: NONE
- ✓ **GENRE**: Idle-Clicker
- ✓ **DESCRIPTION**: Player as an alchemist assistant helps make potions to complete the incoming adventurers' quests

HOW TO PLAY: Player would have to create potions to make money while completing quests (usually by making potions with specific potion levels) from adventurers before the time limit is over (Sinsiroad, 2016)as seen in ***FIGURE 1.06***.

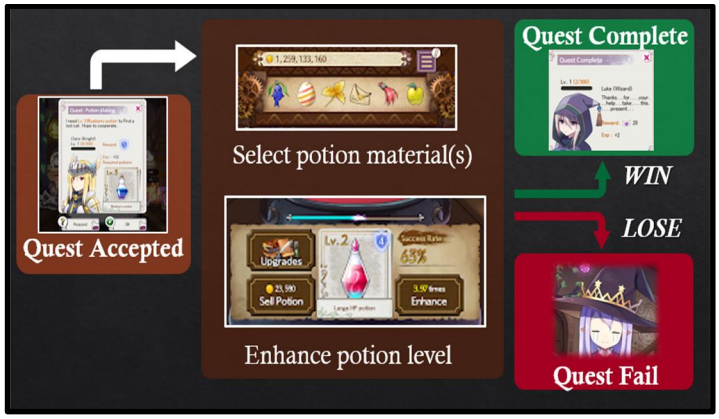


FIGURE 1.06: General Overview of Potion Maker Gameplay

In an example of a quest completion, the player would have to create a potion that has been requested by the client, in which the steps are as shown in ***FIGURE 1.07***.

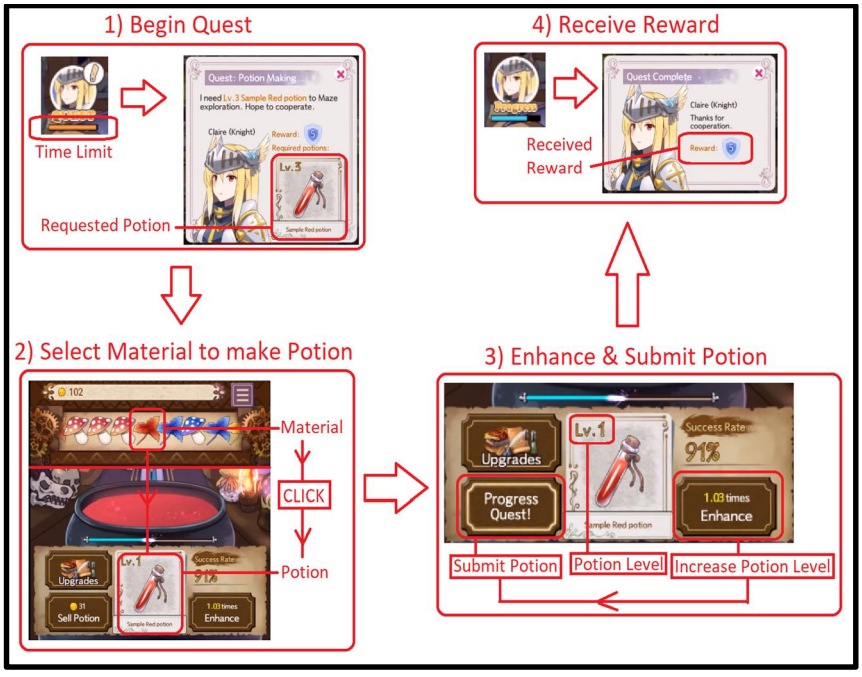


FIGURE 1.07: Sample Gameplay of Quests in Potion Maker

1) Quest is displayed with a time limit on the screen, showing details of rewards and wanted potions (with wanted potion level) when clicked on it.
2) Select suitable material to create the potion.
3) Enhance the potion to desired 'potion level' and send the finished potion to the client to progress quest.
4) Receive the rewards from completing the quest.

WIN CONDITION

- Successfully created the requested potion & send it to the client

LOSE CONDITION (for each quest)

- Time limit for the quest is over

GAME MECHANICS

❖ *CHARACTERS*

➢ Player (invisible in this game)

➢ In-game Avatars (Pio & Tia)

❖ *ITEMS*

➢ Materials & Potions

Material Types	Potions Created	Enhancement Success Rate	Selling Price	Enhancement Amount to Max. Potion Level
Red Items 	Life Potions 	Low	High	Less
Blue Items 	Mana Potions 	Medium	Medium	Medium
Yellow Items 	Stamina Potions 	High	Low	A Lot

➢ Money (usually spent on item OR feature upgrades)

➢ Buffs (potion shield, enhance boost, sale boost)

- **Potion Shield**: Prevents potion from being destroyed when enhance fails
- **Enhance Boost**: Increases rate of success from enhancing potions
- **Sale Boost**: Increases selling price of potions

❖ *ENVIRONMENT*

➢ 2D 'Alchemy Workshop' background with a girl as in-game avatar display

4) Dungeon & Girls: Card RPG

- ✓ **PUBLISHER**: LUNOSOFT INC.
  OTHER GAMES PUBLISHED: River City Ransom: Kunio Returns
- ✓ **GENRE**: Turn-based Card Adventure
- ✓ **DESCRIPTION**: An adventurer who explores in dungeons while recruiting girl partners.

HOW TO PLAY: Progress deeper into the dungeon using a deck of cards & defeat monster girls along the way (LUNOSOFT, 2016) as seen in ***FIGURE 1.08***.

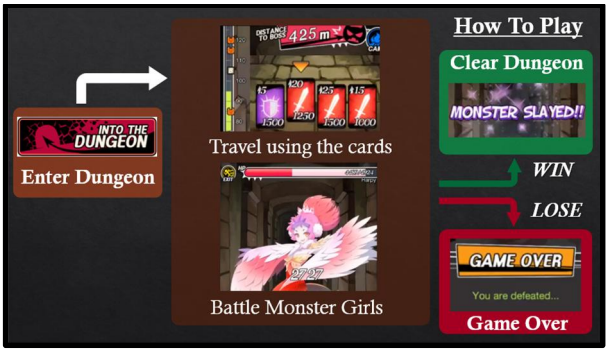


FIGURE 1.08: General Overview of Dungeon & Girls Gameplay

In an example gameplay of a dungeon level, the player would have to reach the end of dungeon using the cards randomly shuffled from the player's deck as shown in ***FIGURE 1.09***.

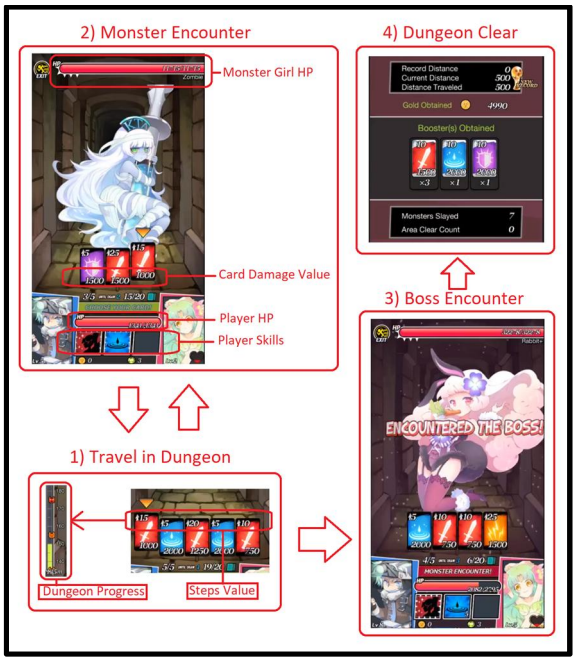


FIGURE 1.09: Sample Gameplay of a Dungeon Level in Dungeon & Girls

1) Use cards to advance through dungeon (card value determines steps taken)
2) When encountering monster girls, use the cards to battle (turn-based format)
3) Progress until reach the end of dungeon
4) Battle BOSS (monster girls too)
5) Defeat boss & clear dungeon

WIN CONDITION

- Completely travel the dungeon
- Defeat the dungeon boss at the end of dungeon

LOSE CONDITION

- When Player HP reaches 0

GAME MECHANICS

❖ *CHARACTERS*

➢ **Player**: Has player skills to venture throughout areas

■ Has different classes that contains different cards ratio (e.g. 1:3:0:2)

➢ **Girl Partner**: Provide player stat boost & extra attack

➢ **Monster Girls**: Attacks player when in contact

■ When defeated, has a chance to join your team as partner

❖ *ITEMS*

➢ Cards

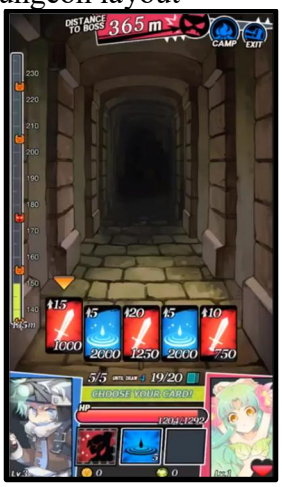
Card Image	Card Type	Effect
	Attack Card	Deals direct damage to opponent (based on the card's base value)
	Counter Card	Reduce damage taken from opponent, and reduced amount would be dealt as direct damage to the opponent
	Heal Card	Recovers player HP (based on card's base value)
	Charge Card	Increases base value of all cards on next player turn only

➢ Player Skills: Varies between the Player's class, but has 3 skill slots each

■ Effects: Extra attack, heal, direct damage, etc.

❖ *ENVIRONMENT*

➢ 2D first-person view dungeon layout



5) Baldi's Basics in Education & Learning

- ✓ **PUBLISHER**: mystman12 (as itch.io game jam submission)
  OTHER GAMES PUBLISHED: NONE
- ✓ **GENRE**: First-person horror
- ✓ **DESCRIPTION**: A student running away from teacher while collecting notebooks to escape from school as shown in ***FIGURE 1.10***.

HOW TO PLAY:

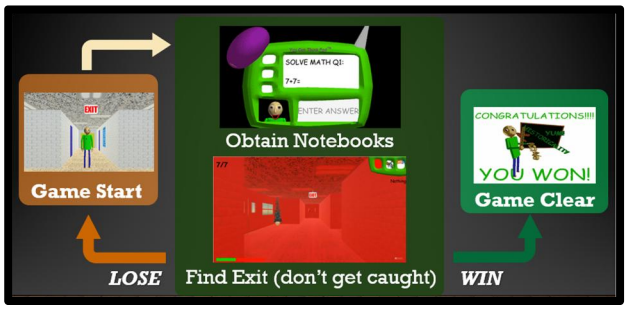


FIGURE 1.10: General Overview of Baldi's Basics in Education & Learning Gameplay

In a demonstration of playing the game, the description of the steps of gameplay would be shown in ***FIGURE 1.11***.

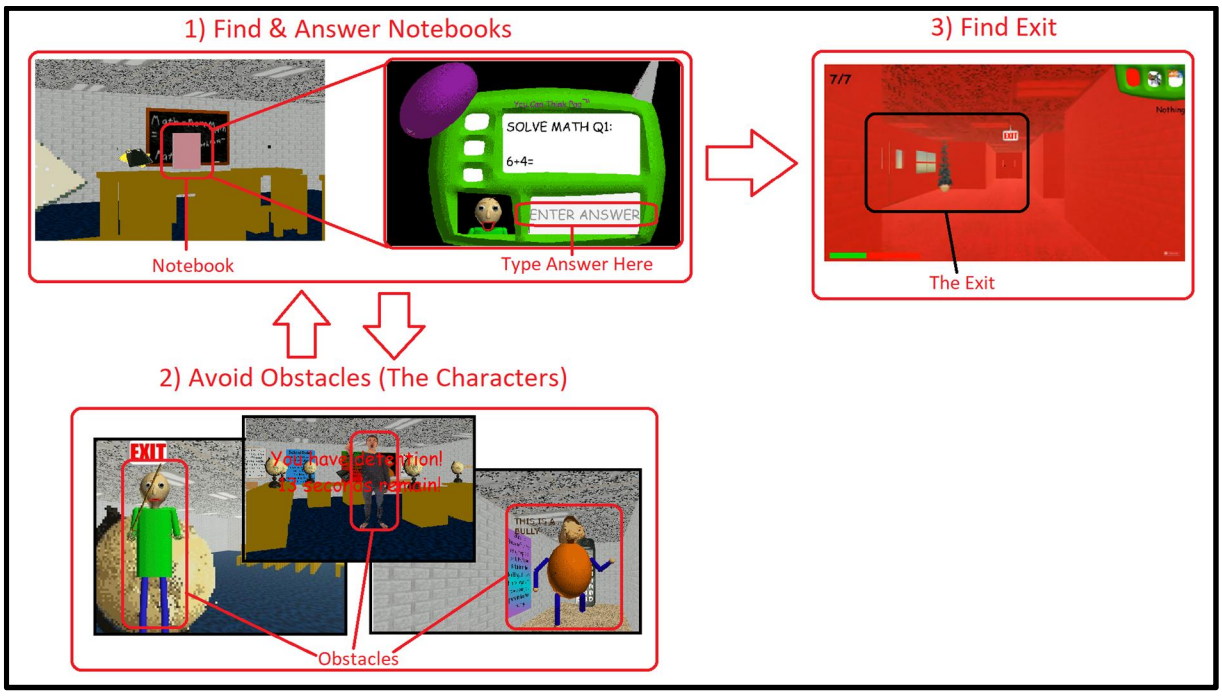


FIGURE 1.11: Sample Playthrough of Bladi's Basics of Education and Learning

1) Find notebooks & Answer the questions in notebooks
2) Avoid any obstacles, including Bladi when collecting all 7 Notebooks
3) Find Exit after finding all 7 Notebooks

WIN CONDITION

- Obtained all 7 Notebooks, answer all questions & find exit (unlocks normal ending)
- Purposely answer all Notebooks' questions **wrong** and find exit (unlocks 'true' ending)

LOSE CONDITION

- When get caught by teacher 'Baldi' (mystman12, 2018)

GAME MECHANICS

❖ *CHARACTERS*

➢ Usually just obstacles, in multiple forms (other than the player)

Character Image	**Character Name**	**Feature(s)**
	Baldi	Instantly triggers GAME OVER when in contact (after answering **wrong** once in the notebook)
	1st Prize	Follows player and pushes any characters in contact
	Playtime	Triggers a jumping minigame that lasts for at least 10 seconds, stopping player's motion in the meantime
	It's a Bully	Blocks the player's path unless it steals 1 item from the player's inventory
	Gotta Sweep	Moves in independent path and pushes anything in contact
	Arts and Crafts	Teleports the player and Baldi to the starting location of the game (school entrance) when in contact. (after player obtains all 7 notebooks)
	Principal of the Thing	Traps player in a 'Detention Room' for 15 seconds whenever it catches the player 'running' in the school.

❖ *ITEMS*

➢ Can be used to stop teacher 'Baldi' from chasing temporarily

➢ Or speed up player movement, reduce/remove obstacles, etc.

❖ *ENVIRONMENT*

➢ 3D environment featuring 'crappy' 90's education programs and applications

Game Reviews: LIM ZHI DA

1) INSIDE

- ✓ **Publisher:** Playdead
- ✓ **Genre:** Puzzle-platformer, adventure
- ✓ **Description:** An adventure of a boy who may run, climb and control object to surpass challenges, but also may die due to different condition (Playdead, 2018).

How to play: Control the character to use/move object to overcome obstacles, avoid the enemy and figure out the way like moving certain object to certain point to proceed further.

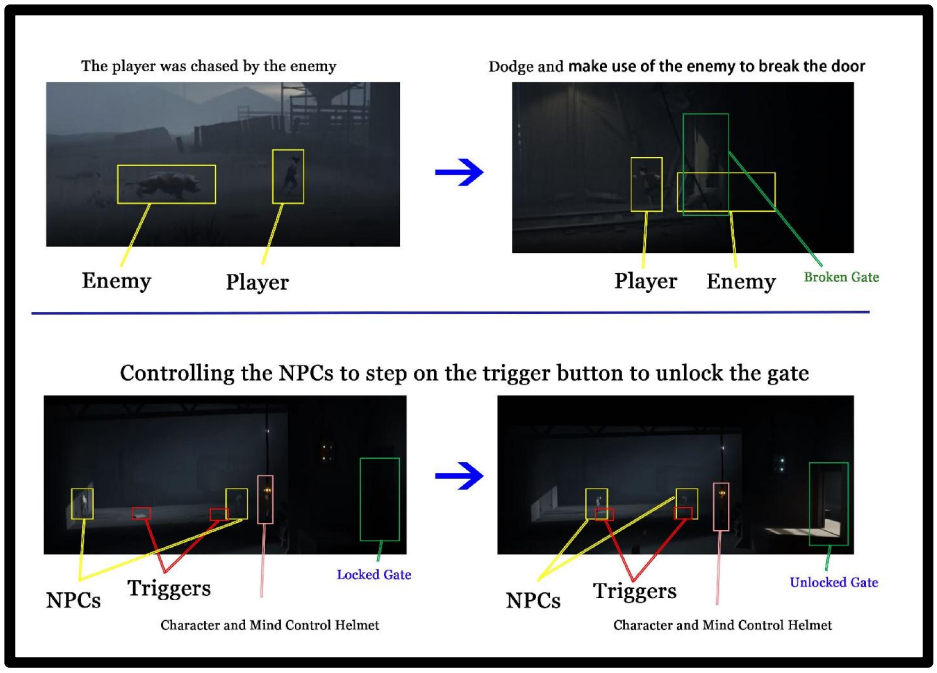


FIGURE 1.12: Showing two examples of solving the puzzles in game

Easter eggs and hidden collectibles are available in the game too, they're not affecting the main gameplay but adding the replay value of the game.

- **Winning Condition:** Escape from the forest successfully.
- **Losing Condition:** Get caught by enemies (dogs, guards etc).

Game Mechanics

- ✓ **Character:** A boy
- ✓ **Item:** Animals, Zombie-like people, Mind-control Helmet
- ✓ **Environment:** A forest with road and incline, Office and Laboratories

2) Samsara

- ✓ **Publisher:** Marker Limited
- ✓ **Genre:** Puzzle-platformer
- ✓ **Description:** Helps an innocent child escape to the destination point, by moving the provided blocks that come with different shape into two mirrored world which separated by the surface of a pool (MarkerLtd., 2018).

How to play: Select, rotate, and place the blocks to create a path to the exit.

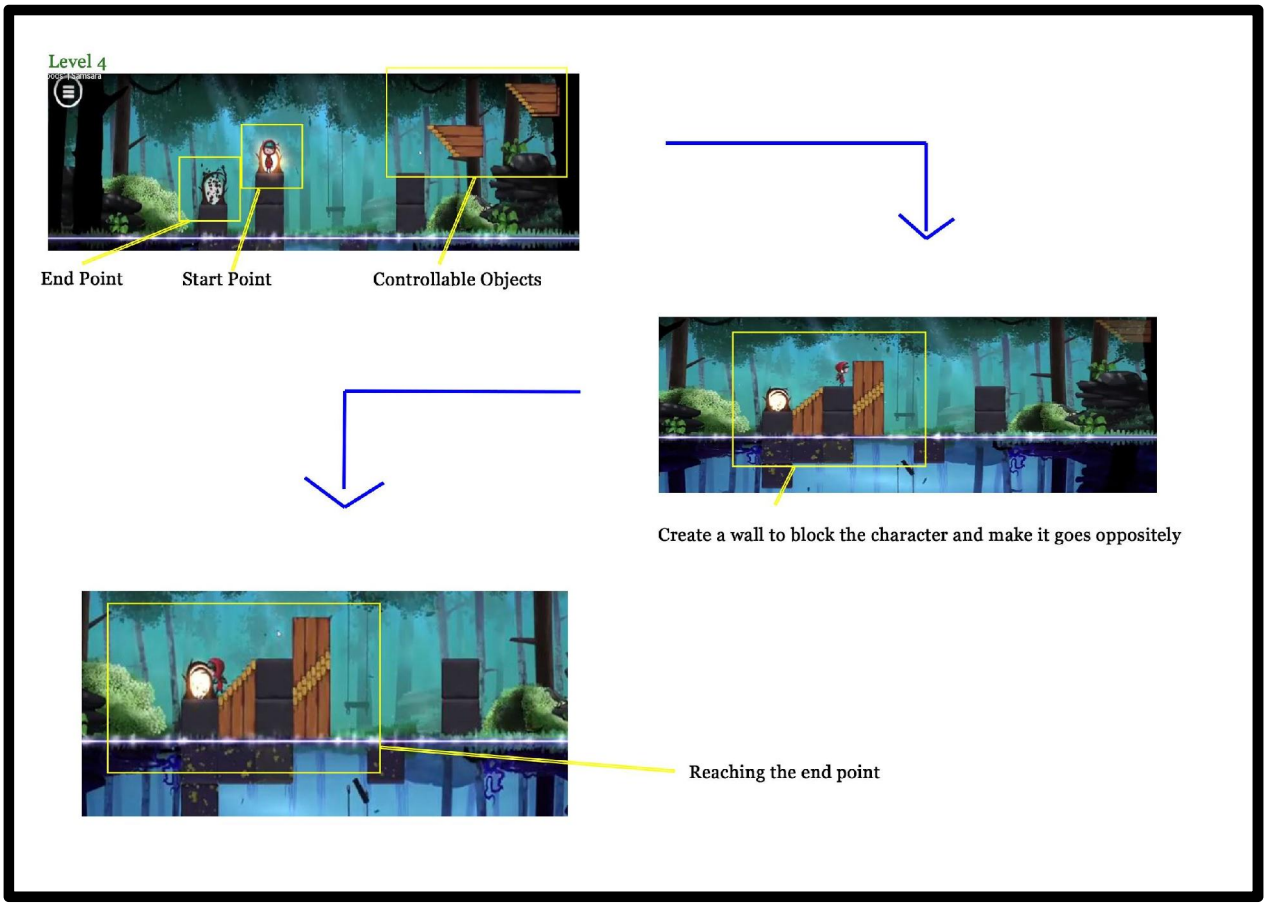


FIGURE 1.13: Samsara Chapter 2 Level 4

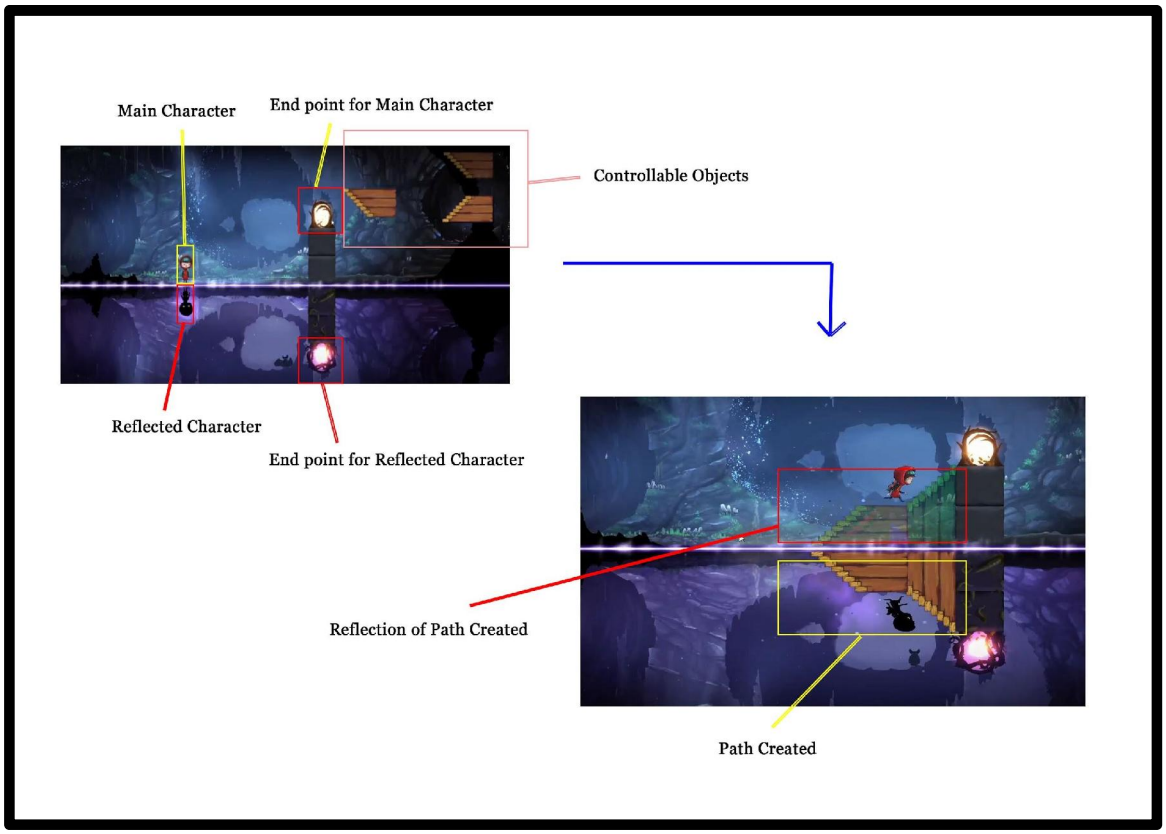


FIGURE 1.14: Samsara Chapter 3 Level 1

- **Winning Condition:** Unlock the correct path for the children to reach the escape point.
- **Losing Condition:** Placed the blocks wrongly, which fails the children from escaping.

Game Mechanics

- ✓ **Character:** Vary kind of Blocks
- ✓ **Item:** Portals, dark root.
- ✓ **Environment:** Mirrored dimensions.

3) Snipperclips

- ✓ **Publisher:** SFB Games
- ✓ **Genre:** Co-operative Puzzle game
- ✓ **Description:** Players control two character which may rotate and moving around the level. One player may snip the overlapping part of another player to create certain shape for solving the puzzle.

How to play: Players may overlap each other to snip the intended shape to move/unlock the puzzle or jump on another player to move to a higher platform (Games, 2018).

- **Winning Condition:** Solve the puzzle.
- **Losing Condition:** Failed to solve the puzzle.

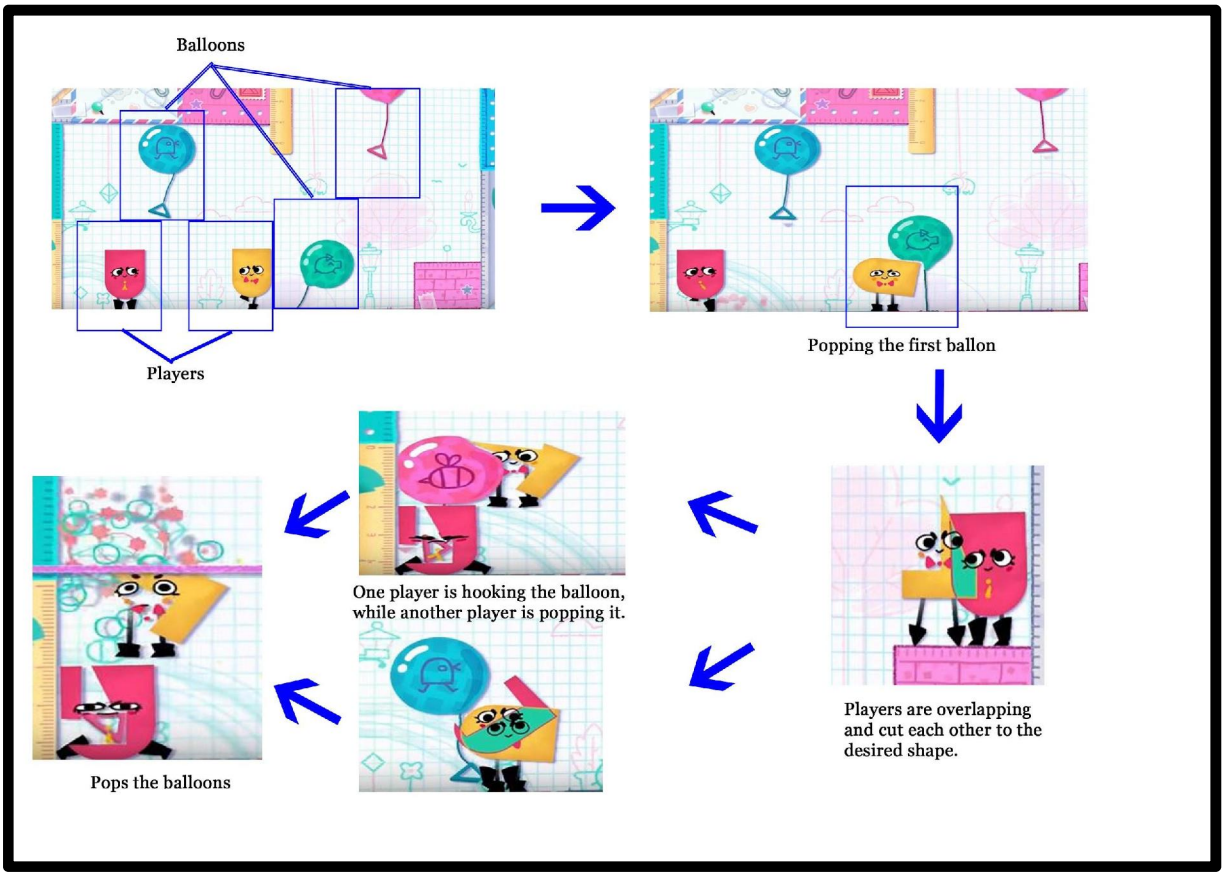


FIGURE 1.15: Level - Balloon Burster

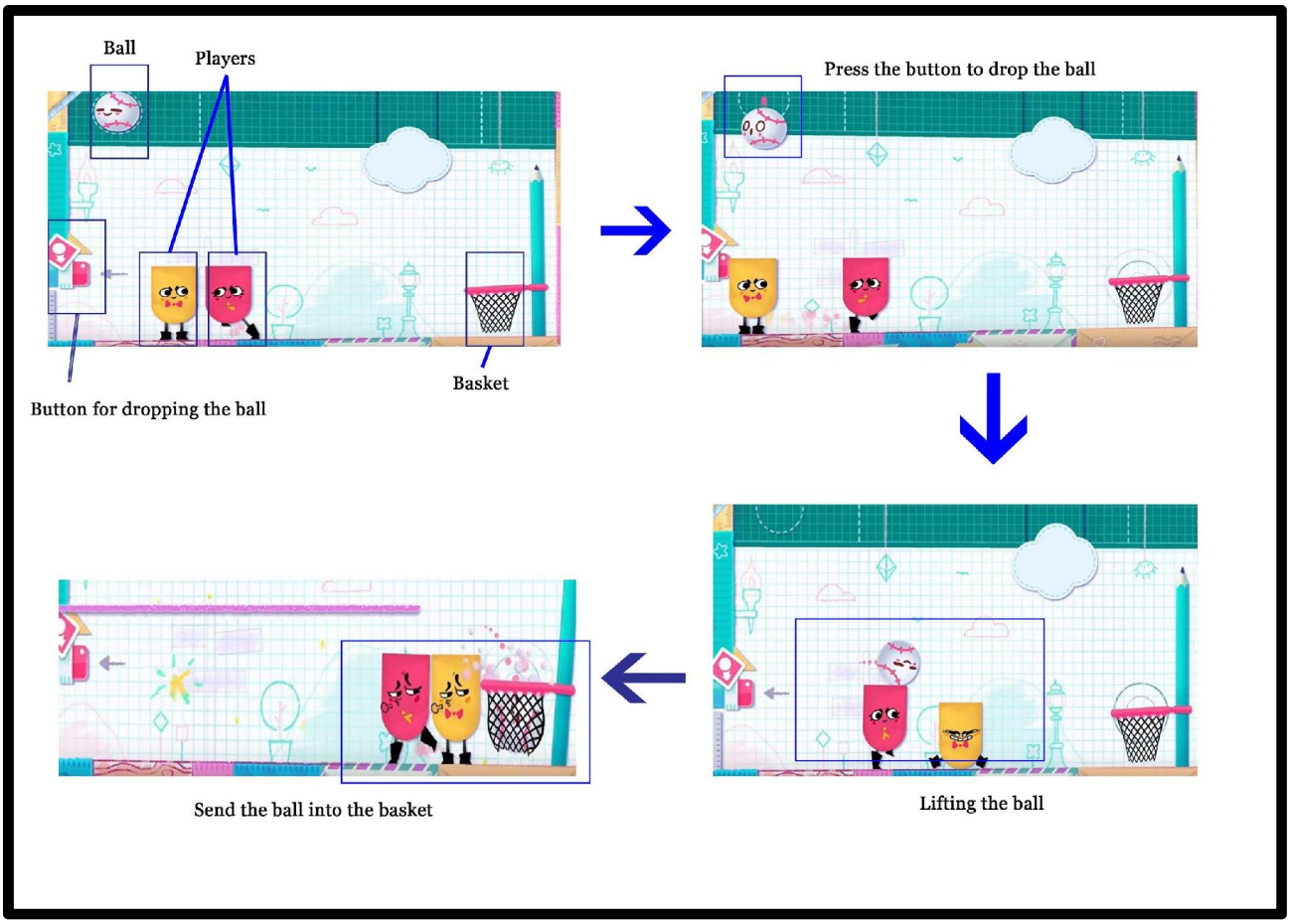


FIGURE 1.16: Level - Baseballin'

Game Mechanics

- ✓ **Character:** Paper Character(s)
- ✓ **Item:** -
- ✓ **Environment:** 2D platform with background that comes along with stationary.

4) *Lara Croft Go*

- ✓ **Publisher:** Square Enix
- ✓ **Genre:** Turn-based puzzle video game
- ✓ **Description:** Move the character turn by turn in the map which is unit based, clearing the level by eliminating enemies or manipulate the scene to solve a certain puzzle (Enix, 2018).

How to play: Controls the character to move or climb in the horizontal plane and vertical platform such as cliff, while the enemies and the environment are also taking the turn to move.

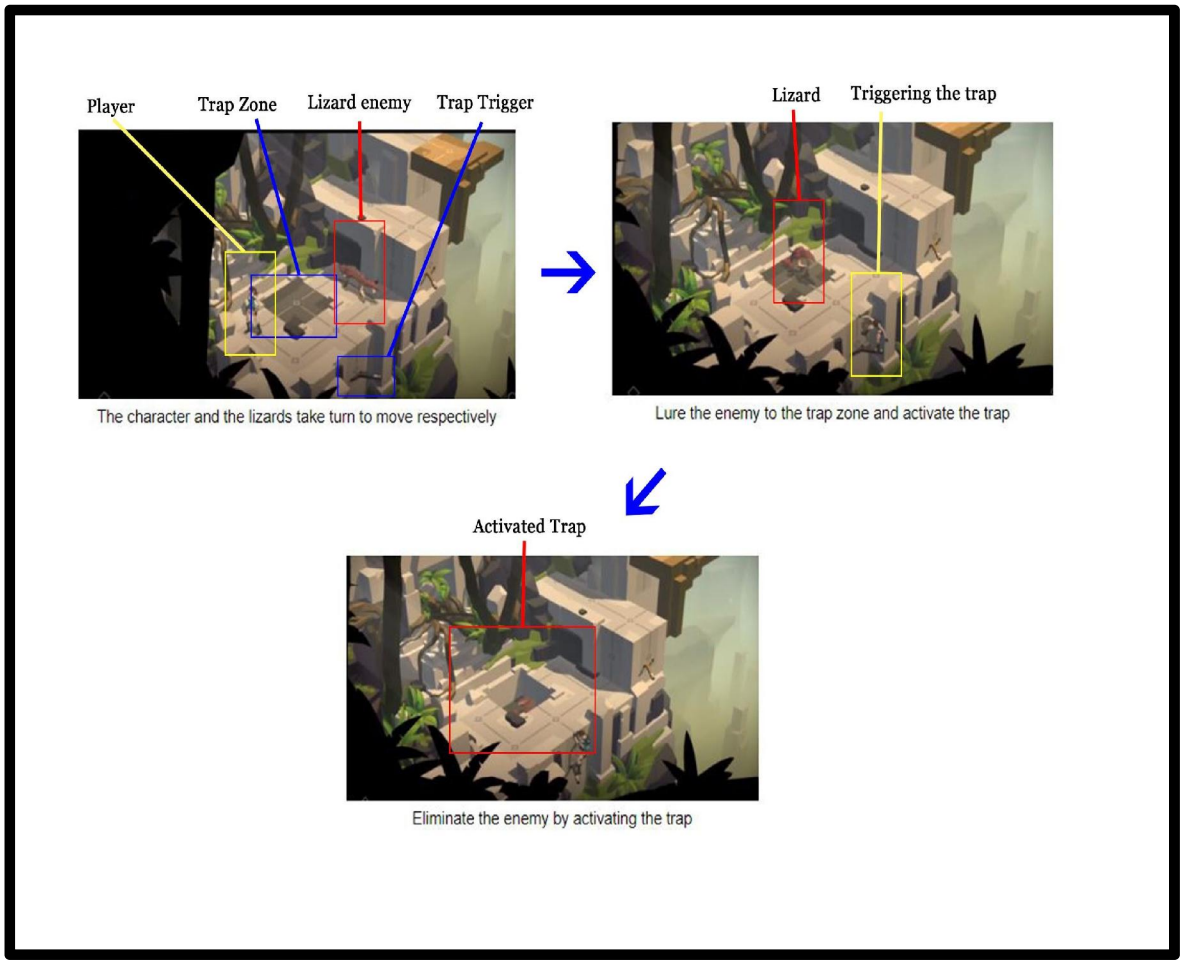


FIGURE 1.17: Puzzle Example 1

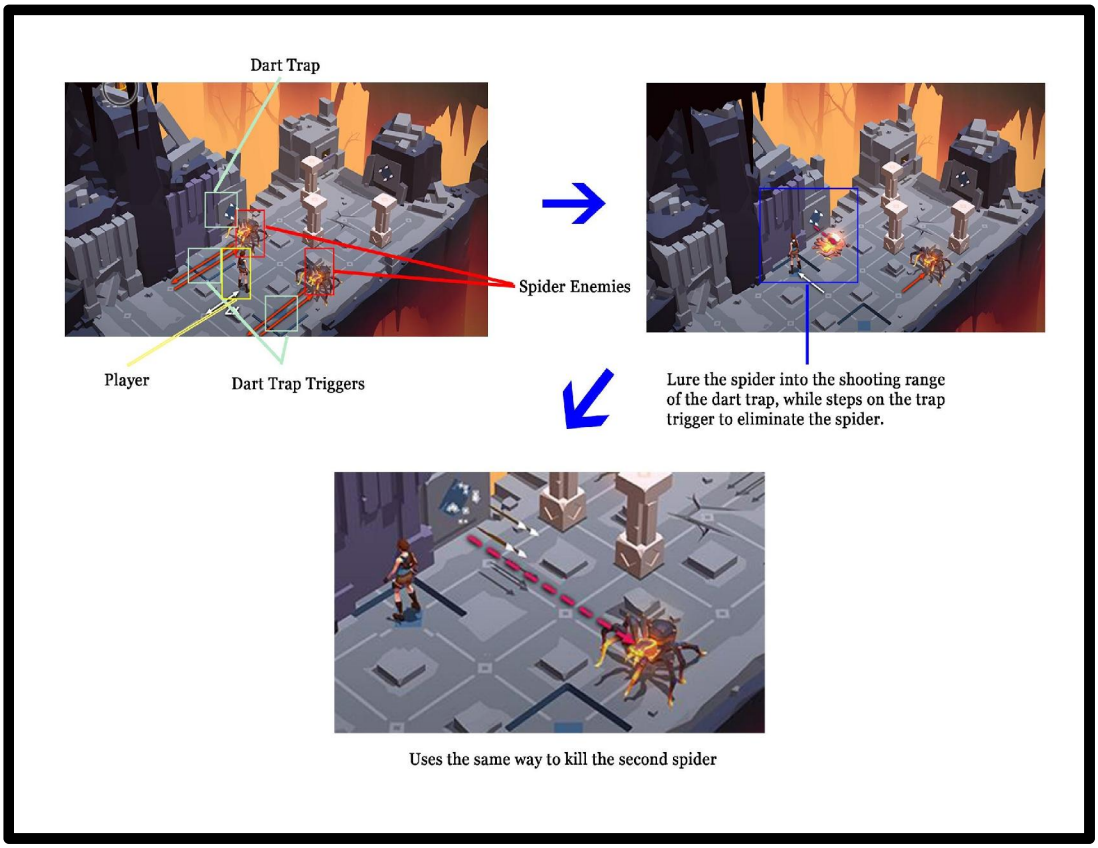


FIGURE 1.18: Puzzle Example 2

- **Winning Condition:** Avoid the enemy and reach the checkpoint of the level.
- **Losing Condition:** Collided with enemy or falls into the trap.

Game Mechanics

- ✓ **Character:** Lara Croft
- ✓ **Item:** Spear, Traps
- ✓ **Environment:** Valley with traps and trap triggers, cliff.

5) The Witness

✓ **Publisher:** Thekla, Inc.
✓ **Genre:** 3D puzzle video game
✓ **Description:** Solves multiple kinds of puzzle and explore the maze in the island.

How to play: Draw a correct path in the puzzle grid to solve the puzzle, usually the hints for the puzzle would appear in the environment.

- **Winning Condition:** Solve the puzzle in the checkpoint and proceed to another section of the island (Thekla, 2018).
- **Losing Condition:** -

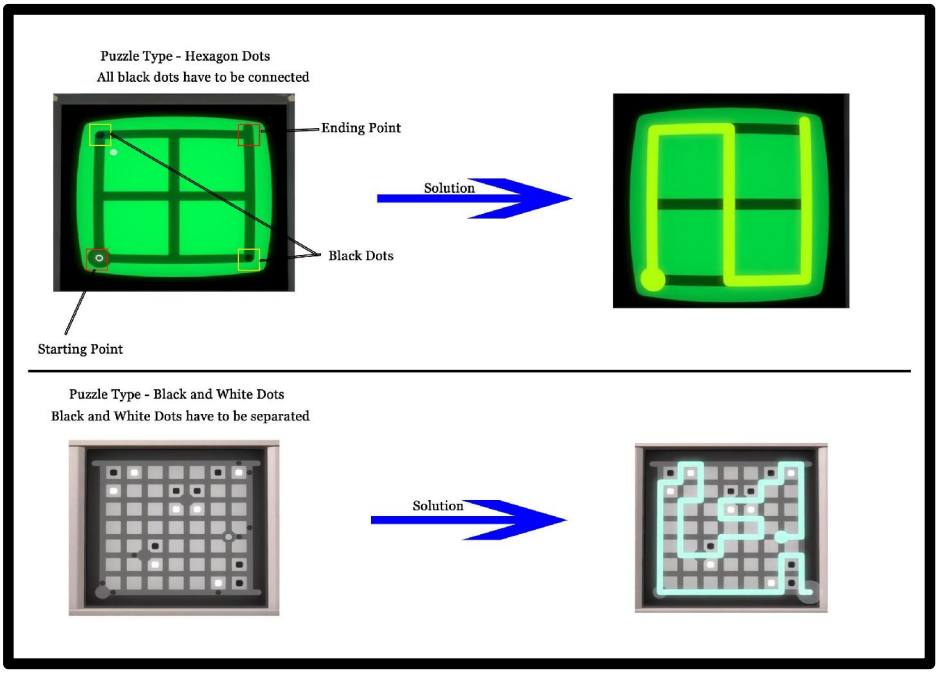


FIGURE 1.19: Puzzle type samples 1 and 2 in The Witness

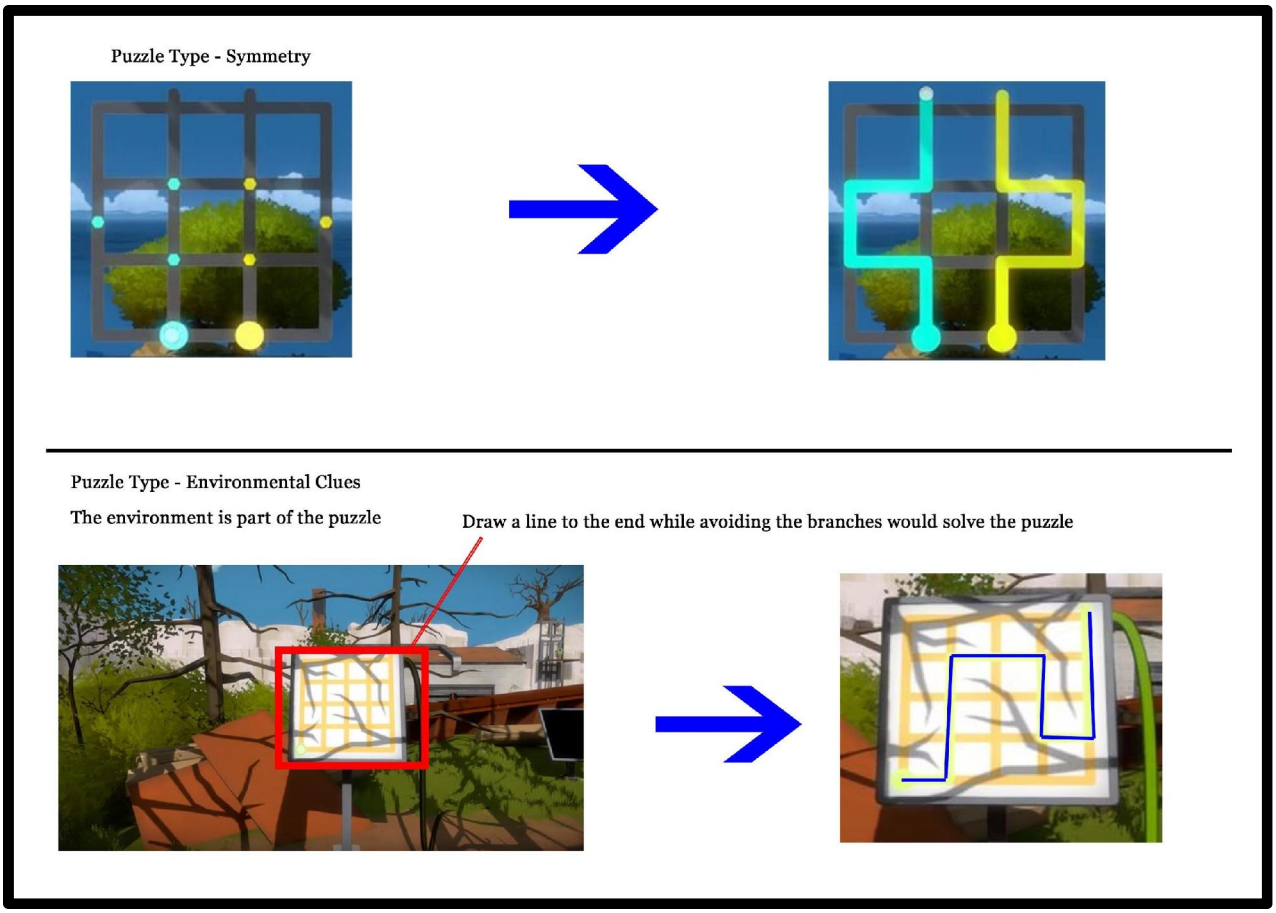


FIGURE 1.20: Puzzle type samples 3 and 4 in The Witness

Game Mechanics

- ✓ **Character:** An unnamed character
- ✓ **Item:** -
- ✓ **Environment:** Island with a lot of locked gate which requires puzzle solving to unlock.

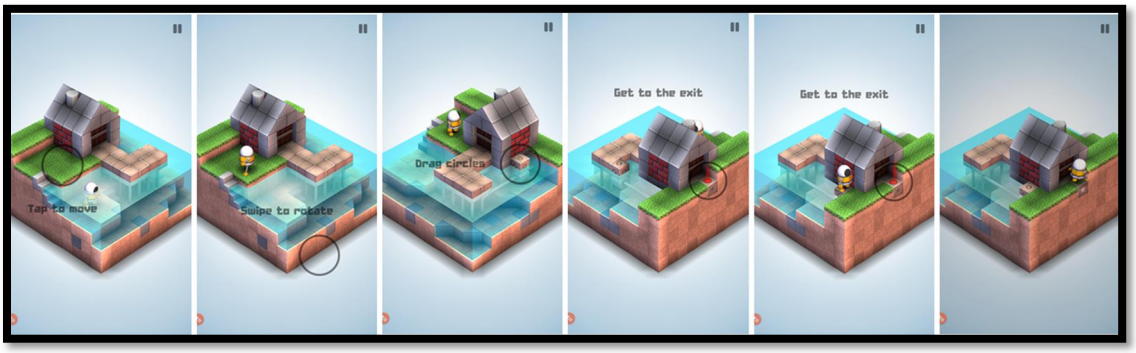
Game Reviews: SEE RONG JIE

1) *Mekorama*

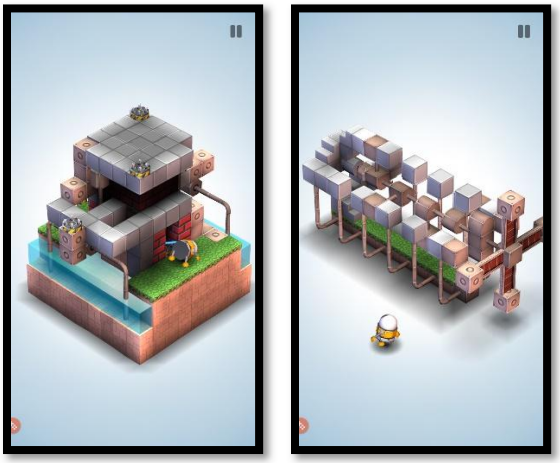
- ✓ **Publisher**: Martin Magni
- ✓ **Genre**: Puzzle
- ✓ **Description**: Help a tiny robot to find home through 50 puzzling mechanics three-dimensional scene (Magni, 2017)

How to play: Swipe left and right to see the scene, tap to let robot move to that point, reach the red dot spot to end a level

Win condition: Guide the robot to reach red end dot.



Lose condition: Restart the stage if end up in these kinds of situation, shocked by electric or leave the puzzle.



Game Mechanics

- ✓ **Character**: Tiny robot and AI robot

- ✓ **Items**: the only movable item is the cube with circle on it

- ✓ **Environment**: 3D environment using blocks to create path, water, traps and buildings. Excellent level design using puzzle elements

2) Monster Strike

- ✓ **Publisher**: Mixi
- ✓ **Genre**: Turn Based Puzzle RPG
- ✓ **Description**: Combat system by using slingshot-style to throw characters in a closed area thus let them bounce around, activate abilities while dealing damage to monsters

How to play: Drag, pull and release the character to bounce and damaged the monsters to clear a stage (Mixi, 2015)

Win Condition:

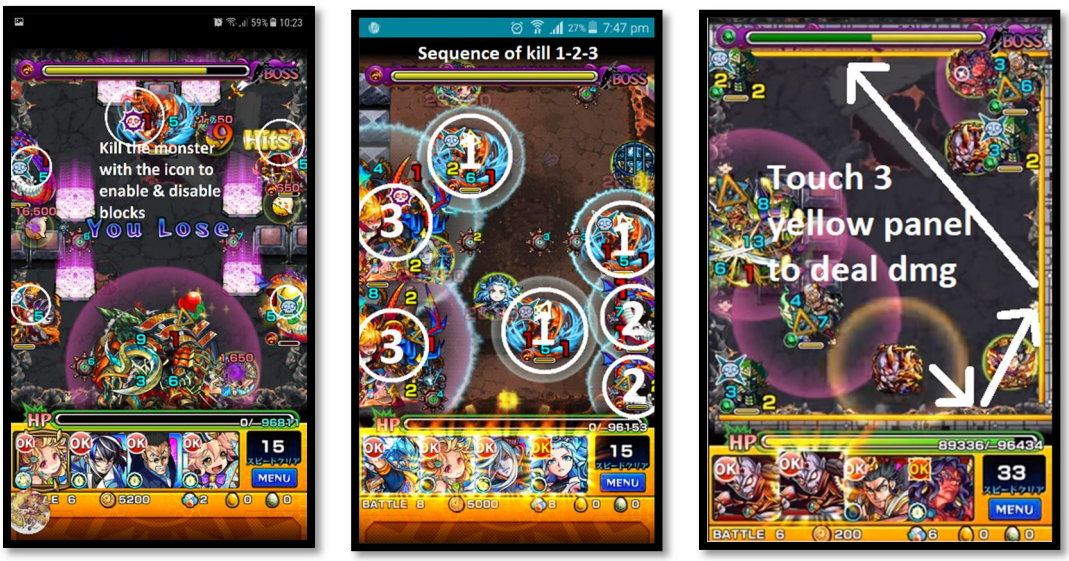
- Each character has their own attack, health, speed, skills, passive ability and active ability
- Choose the marbles that are suitable or counter the stage
- Adjusting the angle and direction to shoot and try to kill the enemies
- Player won when defeated the boss

Lose Condition:

- Health drop to 0

Game Mechanics

- ✓ **Character**: 4 marble characters, monsters
- ✓ **Item**: heart, sword, boots, hourglass, radar etc
- ✓ **Environment**: Rectangle enclosed area and placement of enemies and traps

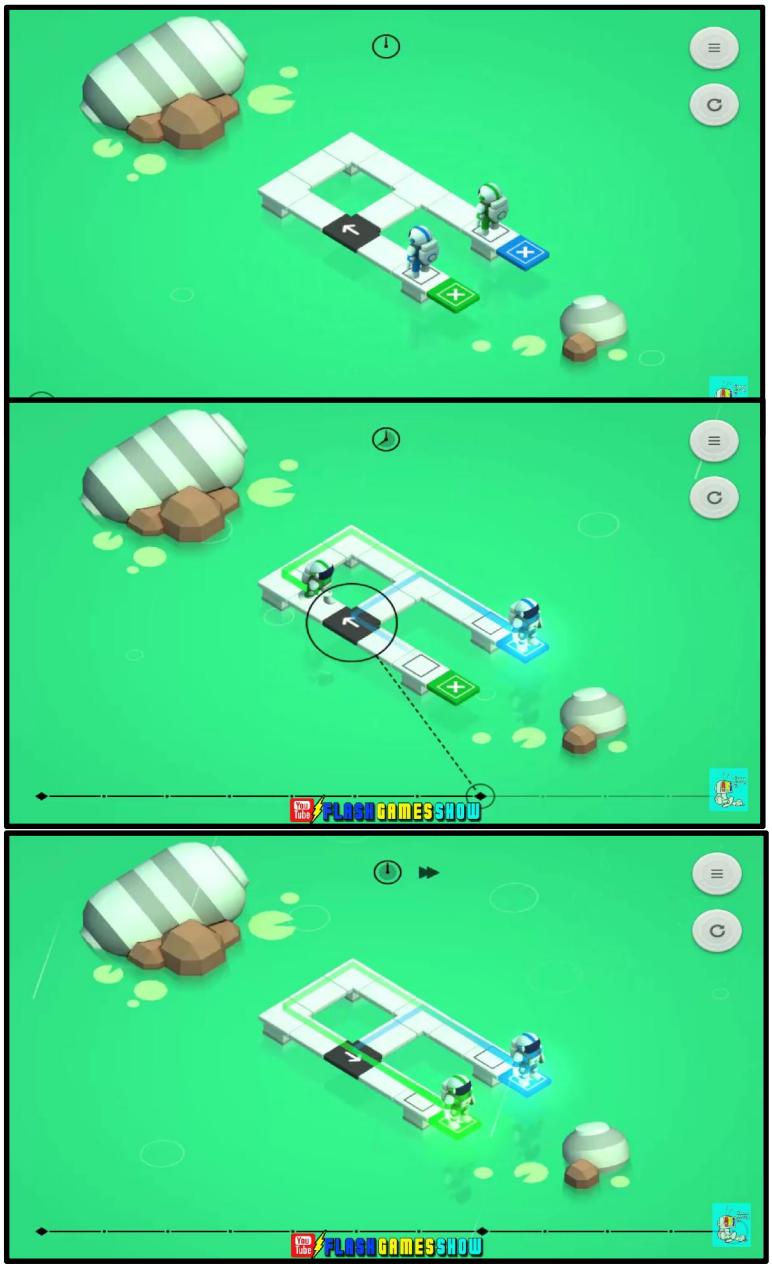


As game goes on, the level will become very difficult as player can only bring some specific characters, even the level has certain sequence to play through or else it cannot even hurt the boss. Moreover, even the angle has to be perfect as well in order to pass the stage.

3) *Causality*

- ✓ **Publisher**: Loju
- ✓ **Genre**: Puzzle
- ✓ **Description**: Help a group of stranded astronauts find a route to safety across alien landscapes, manipulating time and solve paradoxes in this time bending puzzler (Loiu, 2018)

How to play:
Win condition: Guide the astronauts to their respective color panel



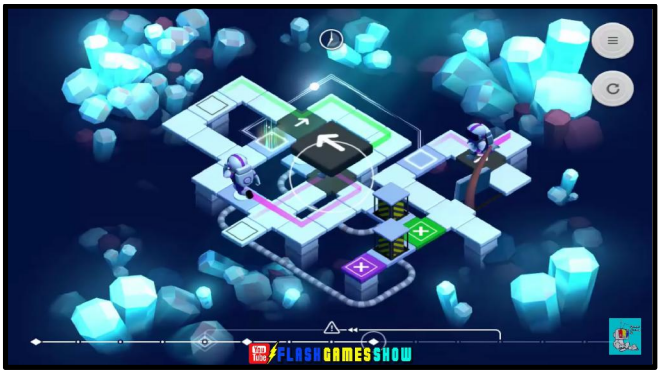
Astronauts will keep on moving, they will change direction only if they hit obstacle.
They will move together at once
Player can control the time by moving back and forward in the timeline below.

Therefore, player can pause the time to change the black arrow panel to guide them.

Since player controls the time, player also can revert back time so basically will not lose.

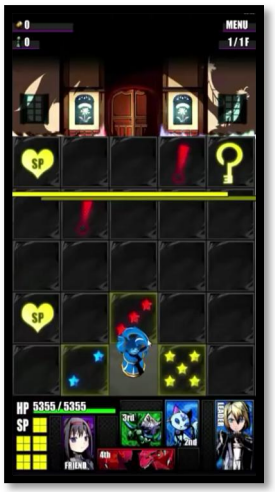
Game Mechanics

- ✓ **Character**: Astronauts with their respective colour on it
- ✓ **Items**: Black direction panels and action timeline
- ✓ **Environment**: 2.5D environment with puzzle path design including alien tentacle, obstacle switches, time travel portal and evil astronauts



4) Divine Gate (Old Version)

- ✓ **Publisher**: GungHo
- ✓ **Genre**: Turn Based Puzzle Strategy RPG
- ✓ **Description**: Combat system by using drag and drop to gather similar colour of stone to a small area to perform attack thus damage the monsters (GungHo, 2016).



Win condition:

Player needs to get the key and move to the door within the 5x5 tiles area

Player can pick which tile they want to step on

Yellow: Safe

Blue: Chance to fight monsters or get trap

Red: High chance to fight monsters or get trap

- Higher Star equal to higher risk or reward

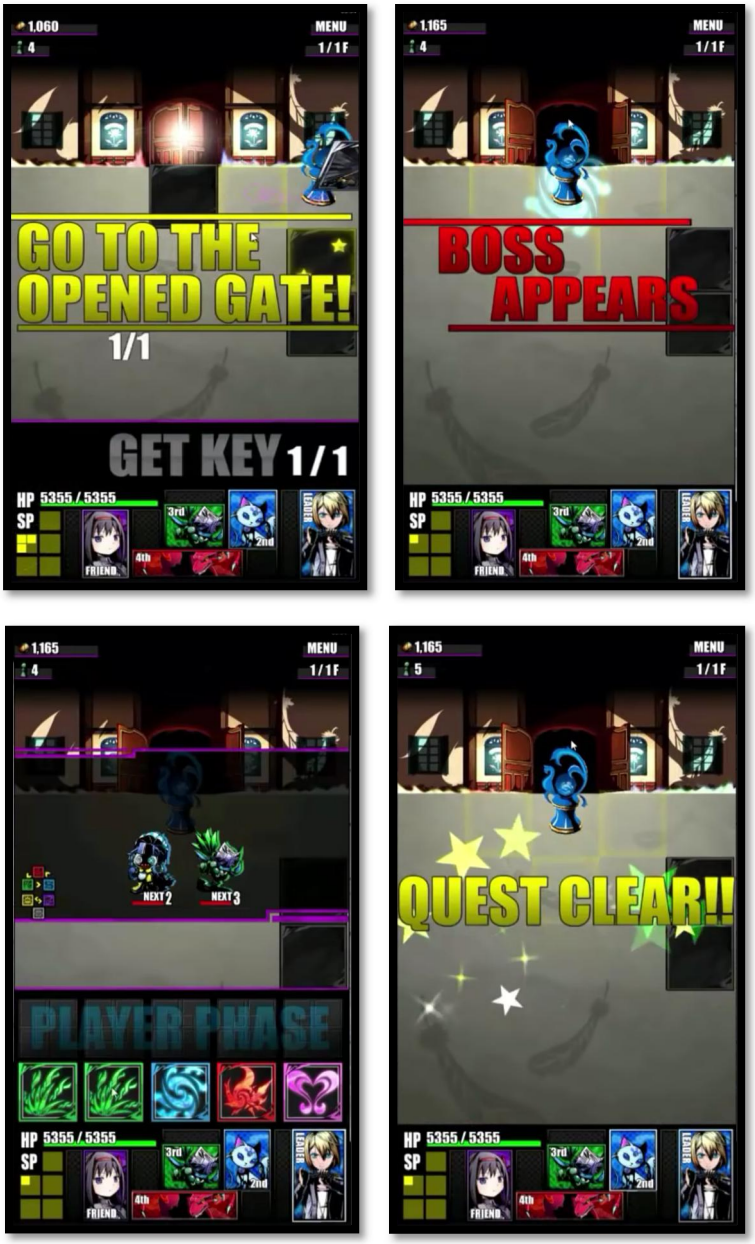
SP: Item drop

Exclamation mark: Random things

Pass through the tiles without letting HP become 0

Monster Appear:

- Total of 5 colors and player has to put 2 or more stones into a box, maximum 5 stones per box
- Player given 5 seconds to complete the action
- Color stone transferred to the characters brought to deal damage to enemies



- Player has to get the key to unlock the door to exit
- Before exit player must deal with the boss first
- After getting the key player can choose to explore the tiles or direct go to the door

Lose condition:

Player's HP become 0

Game Mechanics

Character: Player pieces and 5 characters that players brought to play

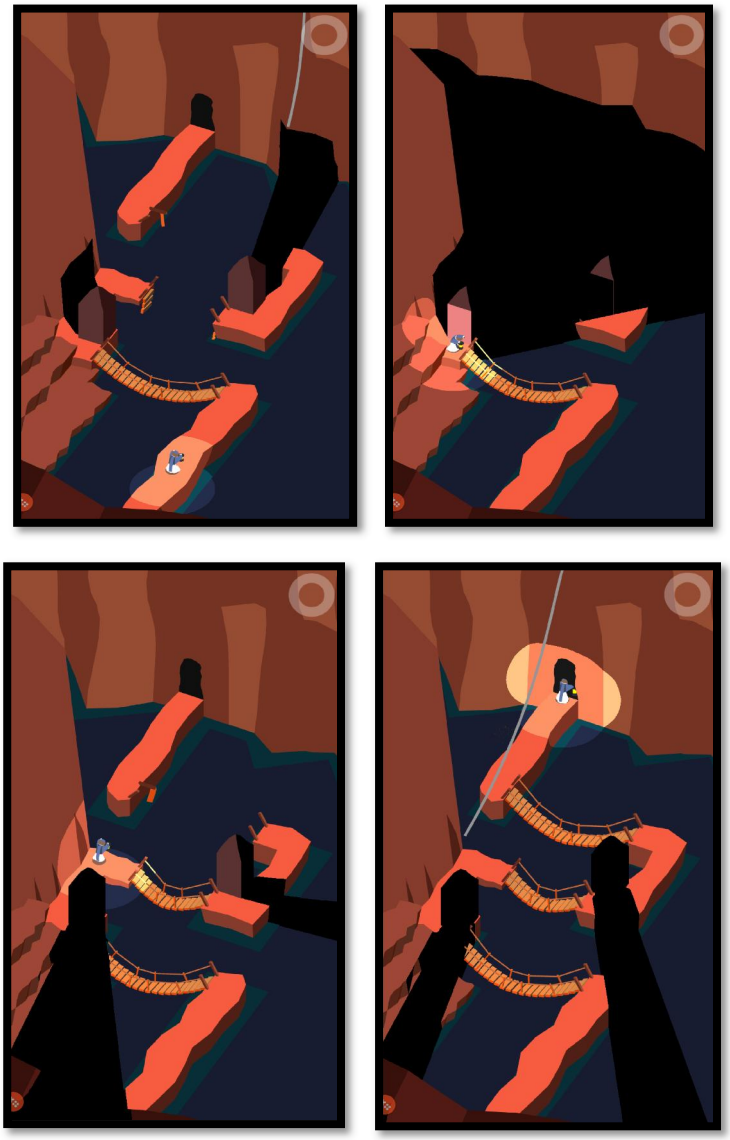
Items: Tiles to be selected and colour stones

Environment: 5x5 tiles, key location and indicator on each tile

5) *Where Shadows Slumber (Demo)*

- ✓ **Publisher**: Game Revenant
- ✓ **Genre**: Adventure Puzzle
- ✓ **Description**: Aid the main character in his search for redemption, in a rule of nature of anything that is not touched by light has freedom to change (Revenant, 2018).

Win condition:
Guide the main character to the cave exit

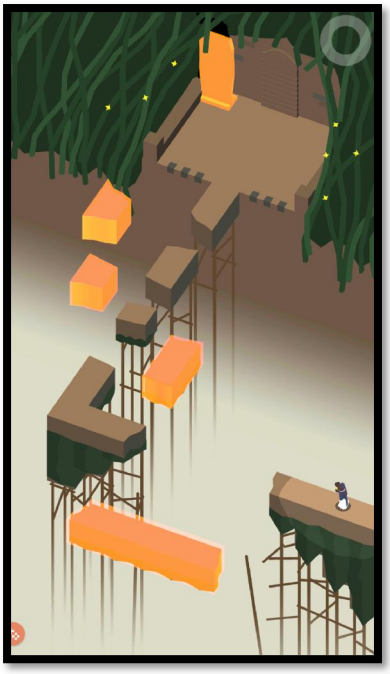


Lose condition:
None, since nothing will attack the player

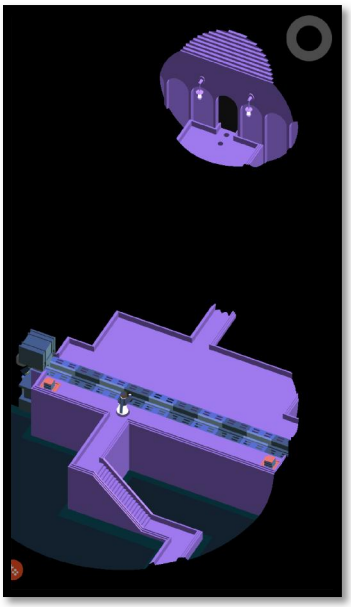
Game Mechanics

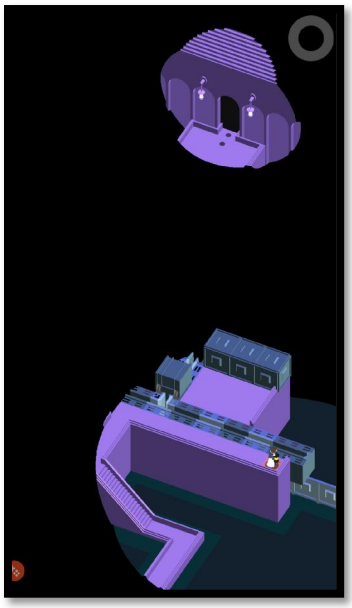
Character: Main character

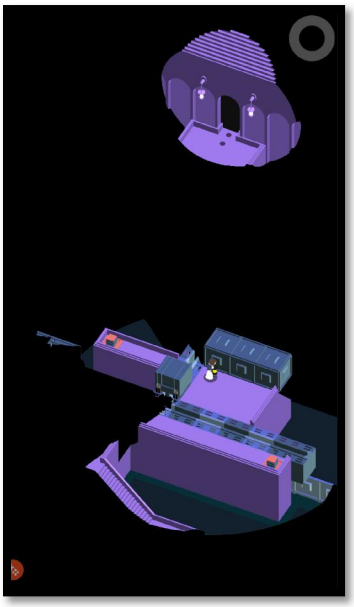
Items: Movable rocks



Environment: Map design, hidden path and buttons







Design & Method

The game planned to produce is a storytelling, semi-roguelike puzzle game with parts of horror elements. For now, its name is set as **“I’m the Skilled Driver”.**

Game Overview

- ✓ Player as **a new cab driver** trying to work for **3 days** to earn money.
- ✓ Player can drive as many times as they like but there is an indicator (progress bar) which shows player's progression per day and **4 customers are available** to choose from.
- ✓ After 3 days of working, your **total income** determines whether you clear the game.
- ✓ If any **fatal accidents** happened to player before last day, it is a GAME OVER
- ✓ If player **did not manage to earn enough money** after last day, it is also GAME OVER
- ✓ GAME CLEAR only happens when **survived all 3 days** and able to **earn enough money** at the end

Level Design

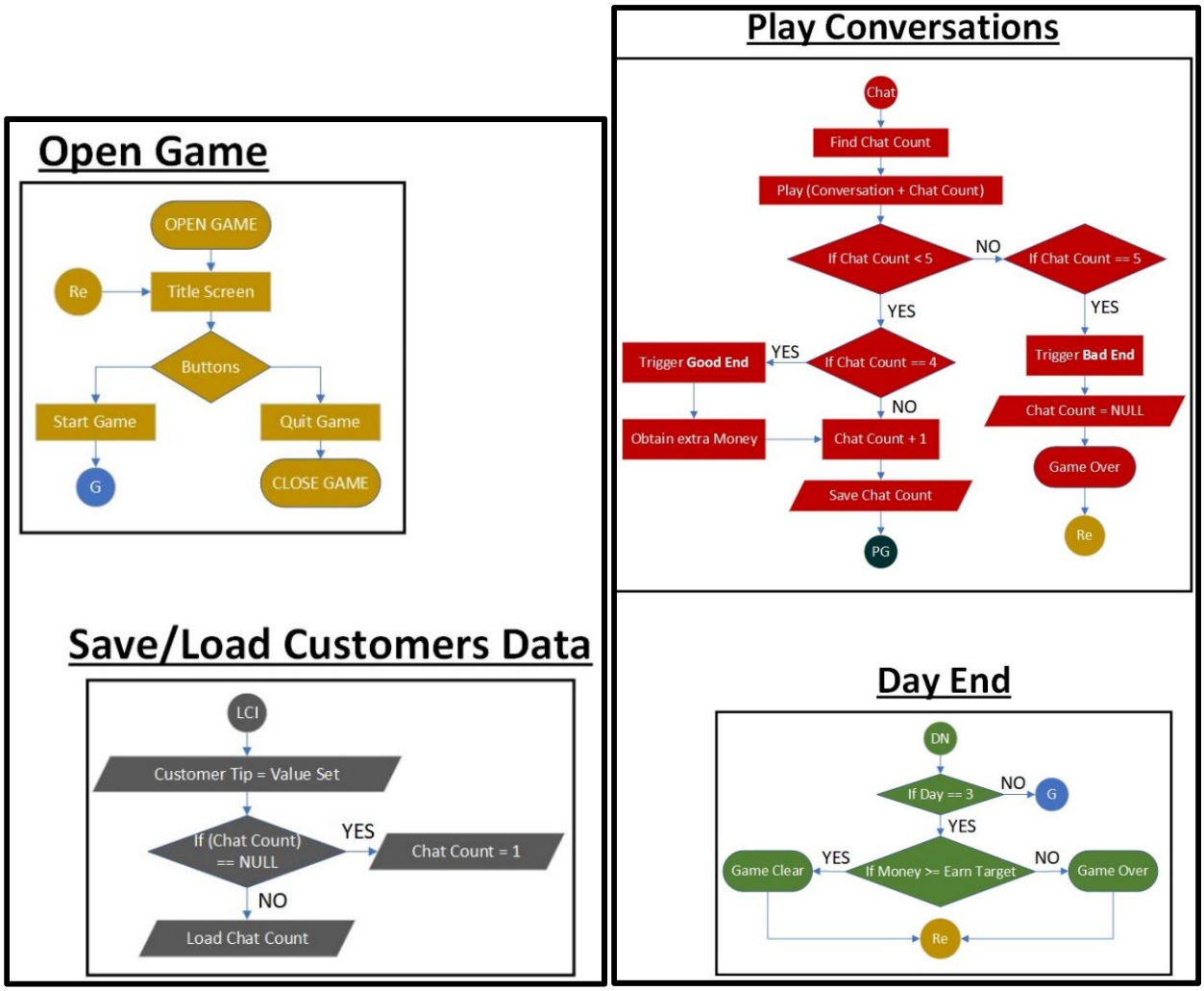


FIGURE 3.2.1: Flow Chart of the Game I

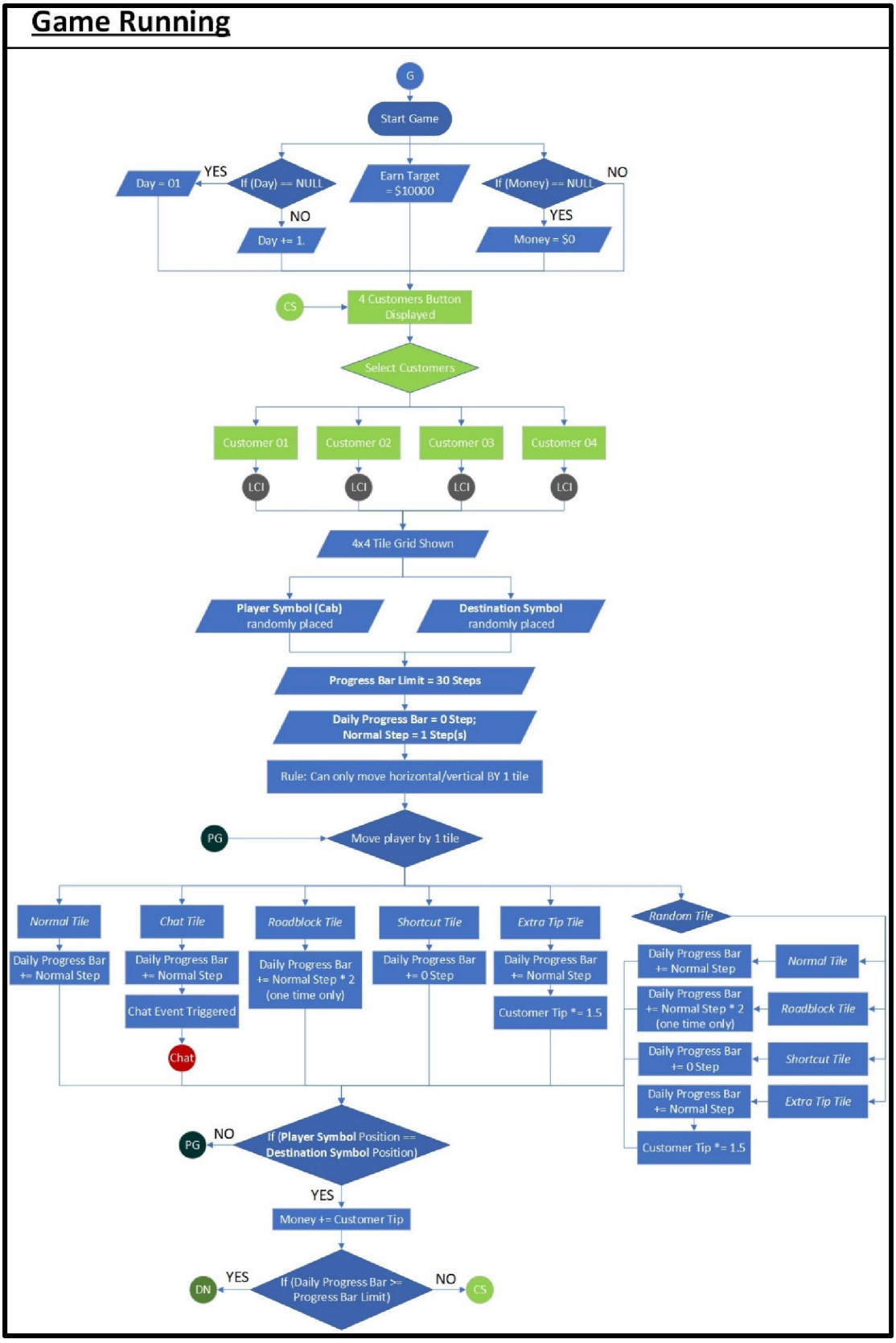


FIGURE 3.2.2: Flow Chart of the Game II

Open Game

Title screen would appear as well as all the buttons UI such as Start Game, Rules, Credits and Quit.

Game Running

Player will be initialized with day 1 and 0 amount of money, player will know that the amount required for 3 days are $10000.

A total of 4 customers will be displayed to let player to choose, a 4*4 tile grid would appear after that as well as the player's car icon. Player only can move horizontal and vertical by 1 tile by swiping it. Progress bar will be shown where it will only increase when player move a step. After 30 steps it goes on to the next day.

Inside the 4*4 tile grid, 6 types of road tiles will appear randomly.

1. **Normal tile**: increase in progress bar by 1 only
2. **Chat tile**: increase in progress bar by 1 thus trigger chat event
3. **Roadblock tile**: increase in progress bar by 2
4. **Shortcut tile**: maintain the progress bar
5. **Extra tip tile**: increase in progress bar by 1 thus increase money get based on customers
6. **Random tile**: randomly choose a tile from above tiles except the Chat tile
7. **Destination tile**: increase in progress bar by 1 thus reload to customers display scene

When the progress bar is full, the player still can continue the game which means that player will not receive any penalty.

Play Conversations

This will only occur when player trigger the Chat Event, the number of chat is calculated and recorded throughout the game. The chat count is increased by 1 each time player passed through the Chat tile. If the chat count increased to 4, player will receive extra money based on the customers. But if the chat count increased to 5, player will reach a Game Over by getting killed by customers.

Save/ Load Customers Data

Each time player triggers the Chat event, the chat count will be increase by 1 and it is recorded with *PlayerPrefs* saving method via the *Unity* editor.

Day End

When day 3 is configured, it will match with the amount earn. If the amount is more than or equal to the requirement, a winning game screen would appear, else a game over screen appear. Next it would appear back to main title screen.

Game Story

Summary

You as a player wanted to earn some money so you tried cab driving as your part-time job for 3 days. You will meet at least 4 people to be fetched as customers during the days but some of them might stop you from earning.

Lore

The game, in actuality, features a murderer who stole a person's taxi and killed the person in the process. The murderer, now the player proceeds to use the taxi to earn some extra money. The taxi, however, was cursed due to the accidents so all customers would gradually become crazier and would kill the player as the victim's revenge.

The murderer, after being killed multiple times by the customers, still go on with using the taxi after every reset. The victim then proceeds to solve the issue herself, personally threatening the player not to 'touch my car' ever again by a jump-scare.

Adapted Concepts

Mobile Input

We've adapted the concepts of using 'Swiping' like *Monster Strike* as mobile input method, as seen from ***FIGURE 3.3.1***. In our script, the minimum swipe distance we set is 10% of the mobile screen height. To implement mobile input, two variables in the format of *Vector3* are needed to be used, while the first one is detecting the position for player's first touch, and the second variable would be detecting the position where the player moved away their finger. The distance of the first touch to the end of the touch would be the touch distance. The system can detect whether the input is a horizontal swipe or vertical swipe by calculating the x axis and the y axis of touch variables mentioned above, after either vertical or horizontal movement is checked, the value of the first touch would be decrease by the value of last touch, if the value of first touch is greater, the movement would be going up in case of vertical movement.

FIGURE 3.3.1: Swipe Tutorial

Roguelike Puzzle

Roguelike Puzzle is the main reference that we have for board generation, usually the turn-based movement like the one in *Dungeon Cards* was utilized in this kind of game. The player moves in turn-by-turn basis, and may trigger different scenarios when approaching special tiles in ***FIGURE 3.3.2***, which means the collision is going to be checked every time the player moves. After the player has cleared the level, a new map should be replaced with different placement of objects.

FIGURE 3.3.2: Roguelike Puzzle

Progress Bar

The use of Progress Bar in *Dungeon & Girls: Card RPG* in clearing dungeons are applicable in setting a time limit for the game's **day calculation**. In *Dungeon & Girls: Card RPG*, the progress bar clearly shows the distance and the steps where in-game benefits and drawbacks are available.

FIGURE 3.3.3: Progress Bar in Game Prototype

While in this mobile game, the progress bar as shown in ***FIGURE 3.3.3*** is used to indicate **the time spent** of driving the taxi in a day. The time spent in progress bar, however, is not clearly indicated to provide a challenge to the player for not having all information laid out and further decrease the game's difficulty.

Horror in Unexpected Segment(s)

In *Baldi's Basics in Education and Learning*, the sudden showing of horror in the cover of an old-fashioned education game is an example of horror in unexpected segments.

FIGURE 3.3.4: Horror segment after talked too much with All Customers

Like ***FIGURE 3.3.4***, this game intends to create the nearly similar effect, where a normal, simple puzzle game that only aims for earning money would be turned into a horror game that gradually limits the player's control on the game when the players talks too much with a specific customer.

System Implementation & Analysis

Created Classes

1. Game Manager & Board Manager

There's only one function for the game manager which is calling in the beginning of the stage, it would initialize and become the **holder of board manager**. *[refer to Appendix A]*

The board manager is utilized for the **creation of the puzzle** in the game. In order to function this class, the number of rows and columns of the puzzle that we want to create have to be determined, in our case, our puzzle map is a '4 x 4' which has **4 rows and 4 columns** respectively.

The board manager is also the container of the tiles that we're going to use, for instance the road tile, roadblock tile, shortcut tile, etc. *[refer to Appendix B]*

The board manager would **generate 16 road tiles** and **a border** which surrounding those road tiles, while the prefab of the player would be created at the position of the first generated road tile, which is the origin point. After the basic layout of the map was generated, the script would **produce the special tiles** such as roadblock tiles, shortcut tiles, the positions for them are randomly selected from one of the road tiles. To organize the generated tiles in *Unity* inspector view, a parent object named *'BoardHolder'* would be spawned and it would collect all the generated tiles.

The two of the most important special tiles would be the *'Chat Trigger'* and the '*Destination'*. The ***'Chat Trigger'* would trigger the specific dialog** with the specific customer selected, the number of the player triggered the 'Chat Trigger' would also affect the content of the dialog. On the other hand, colliding with the ***'Destination'*** tile simply means that the player has successfully reached the destination, the board would be **randomized and generated again**.

2. Player & Flash Trigger

'Player' class serves the role of **player controller** and **determines the event that triggered** by different type of special tiles. From the player controller perspective, player of using computer to launch may control the character by giving the input of 'W, A, S, D' which would move the character to four directions respectively, while for mobile control, we've activated the 'Swipe' for player to control the character. *[refer to Appendix C]*

Particularly for horizontal movement, we've added the function to **flip the sprite to the direction** based on the movement of the character, we're not applying the sprite change for vertical movement is because the sprite looks weird when turning upside down.

By the way, we have added 2 integer variables which are '*blockX'* and '*blockY'* in Player script to **restrict the movement of player**, since our map has only '4x4', we need to stop player from moving out of the boundary. '*blockX'* is used for calculating how many times that the player has moved horizontally, for instance if the player is located at the top bottom corner of the map and it moves to the right once, the '*blockX'* would increase from 0 to 1, and the '*blockX'* would become 3 when the player reached the bottom right of the map. Oppositely, moving towards the left would decrease the '*blockX'* by 1. '*blockX'* should always follow two condition which are '*lower or equal to 3*' and 'larger or equal to 0' to prevent player going beyond the boundary. Same logic has applied with '*blockY'* to maintain the vertical movement.

The *'Flash Trigger'* script is to help player to **reveal the information of tiles** which nearby the character, we've added a cover to hide the details of the tile when the map creation, the player could only discover the tile details only when approaching them. This script is not having the 'Flash' function but the function of destroying the cover of the tile. *[refer to Appendix D]*

3. Level Manager

In *'Level Manager'* class *[refer to Appendix E]*, the class contains the basic variables for the whole game's puzzle mechanic: **Day**, **Money**, **ProgressBar**, and so on. These basic variables are added their relevant upper limits as well to determine **if the conditions are applicable for Game Clear or Game Over**. ***FIGURE 4.1.4.1*** shows the variable storage.

```
public static LevelManager LvMg;
[Header("basic vars")]
public int Day;
public int Money;
public int ProgressBar;
public bool GameClear;
public bool GameOver;

[Header("game main var's limits")]
public int DaysLimit = 3;
public int EarnTarget = 10000;
public int ProgressBarLimit = 30;
```

FIGURE 4.1.4.1: Several Variables in Level Manager

This class also deals with the **basic animations of 3D characters** applicable in the game, where the game would declare the type of animation played by the in-game 3D characters and calling when the characters are supposed to be destroyed after the animation has finished. In the function *dayStart()* shown in ***FIGURE 4.1.4.2***, the animation where the driver (player character) would enter the taxi is declared including the start of the **game's background music**.

```
public IEnumerator dayStart()
{
    dayStartRunning = true;
    yield return new WaitForSeconds(0.5f);
    //spawn driver & driver enters car
    GameObject Driver = Instantiate(driver, driverSpawner.transform.position, driverSpawner.transform.rotation);
    Animator driverAnim = Driver.GetComponent<Animator>();
    driverAnim.Play("DriverEnterTaxi");
    yield return new WaitForSeconds(driverAnim.GetCurrentAnimatorStateInfo(0).length - SoundManager.soundMg.close
    SoundManager.soundMg.sfx_source.PlayOneShot(SoundManager.soundMg.closeCarDoor_sfx, 1.0f);
    yield return new WaitForSeconds(SoundManager.soundMg.closeCarDoor_sfx.length);
    Destroy(Driver);
    SoundManager.soundMg.bgm_source.Play();
    dayStartRunning = false;
}
```

FIGURE 4.1.4.2: dayStart() Function that controls Player

4. Dialogue Manager

'Dialogue Manager' script *[refer to Appendix F]* mains the application of *RPGTalk* plugin, where the talks would be pre-entered into a .txt file as shown in ***FIGURE 4.1.5.1***, and then being callback-ed by this script.

```
[title=Customer1_Talk1_START]
*Customer1*: (whispers) Yes, I got it, I'll be there shortly.
*Customer1*: *beep*
*Customer1*: Excuse me, could you hurry up? I have an urgent meeting soon.
[title=Customer1_Talk1_END]

[title=Customer1_Talk2_START]
*Customer1*: This place can be crowded at times, eh?
*Customer1*: Beware not to bump into any <color=*highlightColor*>roadblocks</color>, lad.
*Customer1*: I don't want to spend more time on traffic than I already do.
[title=Customer1_Talk2_END]

[title=Customer1_Talk3_START]
*Customer1*: It is really often to see you young lads doing this cab driving thing. How much does it earn you?
*Customer1*: If you have time, you should learn some stock investments. That could help you earn your extra kee
[title=Customer1_Talk3_END]

[title=Customer1_Talk4_START]
*Customer1*: Hey, young lad, you listening?
*Customer1*: Apparently the stock of a company I invested in rose up in great numbers.
*Customer1*: How much, you ask? It's higher than you could imagine, lad. Hahahaha.
*Customer1*: Here, take these. I am in good mood today for some extra tips to lads like you.
[title=Customer1_Talk4_END]
```

FIGURE 4.1.5.1: Talk Contents of each Characters for Dialogue Manager Script

This script handles the conversations the player has with other characters by calling a *newTalk()* function with respective functions to play after the talk. Based on the number of talks already initiated, the class would trigger different conversations as shown in ***FIGURE 4.1.5.2***.

```
if (InitiateTalk && LevelManager.LvMg.canTalk)
{
    if (RideCus01)
    {
        Cus1TalkCount = PlayerPrefs.GetInt("C1TC_Temp"); //initial = 1;

        if (Cus1TalkCount == 5)
        {
            SoundManager.soundMg.bgm_source.Stop();
            rpgTalk.NewTalk("Customer1_Talk5_START", "Cus1Talk5_CutScene1Start", rpgTalk.txtToParse
        }
        else
        {
            rpgTalk.NewTalk("Customer1_Talk" + Cus1TalkCount.ToString() + "_START",
                "Customer1_Talk" + Cus1TalkCount.ToString() + "_END",
                rpgTalk.txtToParse, this, "AfterCusTalk");
        }
        InitiateTalk = false;
    }
```

FIGURE 4.1.5.2: newTalk() used in Update() of Dialogue Manager

Implemented Algorithms

1. Algorithms for Board Creation

The purpose of having the board manager class *[refer to Appendix A]* is to create the board in the scene. The map is created by the class in the beginning of the game, by applying the **for loop concept**. The number of the loops depends on the number we set for the 'rows' and 'columns' in board manager class.

For getting the position of each tile, a function named **'Initialise List'** has been created, it would **store each of the position in a list**. For using the position that stored in the list, we implemented the **'Random Position'**, it is basically a function which may **select one position from the Initialise List** for spawning special tile, and **remove the position that has been selected** from the list to avoid the issue of using a same position repeatedly.

A '**Spawn Object**' function has been developed for spawning special tiles, it requires two input which are: which object to spawn and spawn the object for how many times. The function would firstly activate **'Random Position'** to retrieve an available random position, and spawn the object to that position. By calling the function with different input, multiple objects could be created in scene with random position given.

2. Algorithms for Dialogue Sequence

While the dialogues are not intended to be saved manually by the player, the records of the conversation would be recorded in *PlayerPrefs* via integer-type variable. There are 5 *PlayerPrefs* variables that are used to contain these records. When referred to the Dialogue Manager script *[refer to Appendix F],* the *PlayerPrefs* are:

- ✓ CusTalkCount1
- ✓ CusTalkCount2
- ✓ CusTalkCount3
- ✓ CusTalkCount4
- ✓ BadEndCount

While CusTalkCount1, 2, 3, and 4 are used to store the respective conversation amount carried out to each customers, BadEndCount is the variable that is used to contain how many 'Bad Ends' featuring death of player has been carried out throughout the game.

3. Algorithms for Horror Segments

When the variable BadEndCount from Dialogue Manager *[refer to Appendix F]* exceeds 4 times, a 'mysterious customer' would be the only customer of selection, as shown in the *Update()* of Level Manager script [refer to Appendix E]. The game would then proceed to become 'unplayable', where the game does not allow player to accept any input in-game.

The stalemate, however, can be resolved when pressing a hidden 'ReStart' button shown above the game's title which could only appear when the variable BadEndCount reaches the value of more than or equal to 5.

User Manual/Guide

It is recommended that players start by reading the **RULES** section of the game first before entering the game. As mentioned in the rules, player need to tap to choose a customer which they wanted to fetch. In order to move the player's car, they have to swipe it. Please take noted that progress bar raises for each move. The goal of the game is to earn the $10000 within 3 days and the lose condition would be either insufficient of amount at the last day or trigger any bad ends. Lastly, player is reminded to recognize all the icons so that player can have smoother progress.

Test Cases

There are at least 3 people were invited as tester to the game after finished developing. Most of them are performing Functionality Testing to identify bugs and errors that may affect the user-experience. The key features of this testing is to determine whether the application is working accordingly to the specifications which is considered black-box testing. Tester took their time to look for game play issues, graphic issues, audio issues and many more. Even the process of installation is tested.

Besides, Play Testing is also used for the project. The aim is to analyses non-functional features such as fun factors, difficulty levels and balance. Any reviews or comments are welcomed. The key features is to judge the game rather than the facts and to check the game is working in a well-structured manner.

The results of the testing is considered satisfying as the bugs and errors that occur are quite minor. Bugs do appear especially for the chat system and puzzle elements. The requirement for the money amount is being adjusted to balance the game. Other than that, the game runs well as per expected.

Conclusion

Throughout this assessment, a mobile application featuring a semi-roguelike puzzle game with parts of horror elements of a taxi driver is created. The game is made with the game environment incorporated with the 3D objects, mixed with 2D sprites that act as user interfaces that affect the animations of the 3D objects in the game.

In the process of developing the output, the main objectives of creating a mobile application game with a puzzle theme and contains level continuity have been created. The creation of the game prototype starts with the PC (Windows) format and was then converted into the Android build. While the puzzle theme has been included, the puzzle segment of the game is not the focus, but the hidden horror segment of the game. The level of continuity for this game is complete from the beginning till the end, which ensures the puzzle element would not be entirely similar per game.

While we are decently satisfied with the output, there were a few adjustments that could be made given the opportunity and time. First, the animation was made simple due to the restriction of the mobile processing performance for our testing device, and thus several animation sequences were to be cut short. The second improvement could be made via the mobile inputs. While swiping feature works on the game well, it would be recommended to work on a simpler input for the players in the future, like a touch-and-release input.

Workload Matrix

WEIGHTAGE	**ANG CHEE SIAH (TP038259)**	**LIM ZHI DA (TP041644)**	**SEE RONG JIE (TP038306)**
Level Design	30%	35%	35%
Game Story Design	50%	25%	25%
Asset Creation	15%	15%	70%
Programming (Story & Dialogues)	50%	30%	20%
Programming (Puzzle)	25%	60%	15%
Documentation	30%	35%	35%
TOTAL WEIGHTAGE	**33.33 %**	**33.33%**	**33.33 %**
SIGNATURE			

References

1) Enix, S., 2018. *Steam | Lara Croft GO.* [Online]
   Available at: https://store.steampowered.com/app/540840/Lara_Croft_GO/
   [Accessed 10 July 2018].
2) Games, S., 2018. *Nintendo Games: Snipperclips.* [Online]
   Available at: https://www.nintendo.com/games/detail/snipperclips-switch
   [Accessed 3 July 2018].
3) GungHo, 2016. *QooApp | Divine Gate.* [Online]
   Available at: https://apps.qoo-app.com/en/app/1733?from=search:q%3DDivine%2520Gate%26source%3D-1
   [Accessed 5 July 2018].
4) Loiu, 2018. *Google Play | Causality.* [Online]
   Available at: https://play.google.com/store/apps/details?id=com.lojugames.android.Causality&hl=en
   [Accessed 30 June 2018].
5) LUNOSOFT, 2016. *Google Play | Dungeon & Girls: Card RPG.* [Online]
   Available at: https://play.google.com/store/apps/details?id=com.iqubi.dungeonandgirlsunity&hl=en
   [Accessed 30 June 2018].
6) Magni, M., 2017. *Google Play | Mekorama.* [Online]
   Available at: https://play.google.com/store/apps/details?id=com.martinmagni.mekorama&hl=en_US
   [Accessed 27 June 2018].
7) MarkerLtd., 2018. *Steam | Samsara.* [Online]
   Available at: https://store.steampowered.com/app/652880/Samsara/
   [Accessed 26 June 2018].
8) Mixi, 2015. *QooApp | Monster Strike.* [Online]
   Available at: https://apps.qoo-app.com/en/app/1498/%E3%83%A2%E3%83%B3%E3%82%B9%E3%82%BF%E3%83%BC%E3%82%B9%E3%83%88%E3%83%A9%E3%82%A4%E3%82%AF
   [Accessed 23 June 2018].
9) mystman12, 2018. *itch.io | Baldi's Basics in Education and Learning.* [Online]
   Available at: https://mystman12.itch.io/baldis-basics
   [Accessed 30 June 2018].
10) Pixels, 7., 2018. *Google Play | Dungeon Cards.* [Online]
    Available at: https://play.google.com/store/apps/details?id=com.The717pixels.DungeonCards&hl=en
    [Accessed 22 June 2018].
11) Playdead, 2018. *Inside | Playdead.* [Online]
    Available at: http://www.playdead.com/games/inside/
    [Accessed 24 June 2018].
12) Revenant, G., 2018. *Google Play | Where Shadows Slumber.* [Online]
    Available at: https://play.google.com/store/apps/details?id=com.jfmc.extinguish
    [Accessed 11 July 2018].

13) shiba, m., 2017. *1BeatHeart.* [Online]
Available at: http://1beatheart.nobody.jp/
[Accessed 17 June 2018].
14) Sinsiroad, 2016. *Google Play | Potion Maker.* [Online]
Available at:
https://play.google.com/store/apps/details?id=com.sinsiroad.potionmaker&hl=en
[Accessed 25 June 2018].
15) Thekla, 2018. *Steam | The Witness.* [Online]
Available at: https://store.steampowered.com/app/210970/The_Witness/
[Accessed 15 July 2018].
16) vgperson, 2017. *1BeatHeart | vgperson translation.* [Online]
Available at: http://vgperson.com/games/1beatheart.htm
[Accessed 17 June 2018].

Appendix A: Game Manager Script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class GameManager : MonoBehaviour {

	public BoardManager boardScript;
	public static GameManager managerInstance = null;

	#region SceneSetup
	// Use this for initialization
	void Awake () {
		if (managerInstance == null)
			managerInstance = this;
		else if (managerInstance != this)
			Destroy(gameObject);

		boardScript = GetComponent<BoardManager>();
		InitGame();
	}

	public void InitGame()
	{
		boardScript.SetupScene();
	}
	#endregion


}
```

Appendix B: Board Manager Script

```
void InitialiseList()
{
	gridPositions.Clear();

	for (int x = 0; x < columns; x++)
	{
		for (int y = 0; y < rows; y++)
		{
			gridPositions.Add(new Vector3(x+initial_XOffset, y+initial_YOffset, 0f)); //creating possible positions to place object
		}
	}
}

void BoardSetup()
{
	boardHolder = new GameObject("Board").transform;

	//create a boarder surrounding all tiles
	for (int x = -1; x < columns + 1; x++)
	{
		for (int y = 0; y < rows; y++)
		{
			GameObject toInstantiate = roadTiles[UnityEngine.Random.Range(0, roadTiles.Length)]; //randomly select one of the sprite from the list
		//toInstantiate.GetComponent<SpriteRenderer>().color = Color.black;
		GameObject roadCover = Instantiate(surfaceCoverTiles, new Vector3(x + initial_XOffset, y + initial_YOffset, -1f), Quaternion.identity);
		roadCover.transform.SetParent(boardHolder);
		if (x == -1 || x == columns)
			{
				toInstantiate = outerWallTiles[UnityEngine.Random.Range(0, outerWallTiles.Length)];
			}

			GameObject instance = Instantiate(toInstantiate, new Vector3(x+initial_XOffset, y+initial_YOffset, 0f), Quaternion.identity) as GameObject;

			instance.transform.SetParent(boardHolder);
		}
	}
```

```
  public void SetupScene()
  {
	StartCoroutine(SetupSceneCoroutine());
	SpawnPlayer();

	board = GameObject.Find("Board");
   board.SetActive(false);

	playerP = GameObject.Find("PlayerPrefab(Clone)");
   playerP.SetActive(false);

   Debug.Log("setupscene()-> board.setActive false");
}

  public IEnumerator SetupSceneCoroutine()
  {
	BoardSetup();
	InitialiseList();
	SpawnObject(destination, 1);
	SpawnObject(shortcutTiles, 2);
	SpawnObject(roadBlocksTiles, 2);
	SpawnObject(extraTipsTiles, 2);
	SpawnObject(randomEventsTiles, 1);
	SpawnObject(speechBubbleTiles, 2);
	yield return null;
  }
```

```
public void SpawnPlayer()
{
	playerHolder = Instantiate(playerPrefab, new Vector3(0+initial_XOffset, 0+initial_YOffset, -0.0002f), Quaternion.identity) as GameObject;
}

public void ResetBoard()
{
	playerHolder.transform.position = new Vector3(0 + initial_XOffset, 0 + initial_YOffset, -0.0002f);
	Player.playerInstance.blockX = 0;
	Player.playerInstance.blockY = 0;
	Player.playerInstance.GetComponent<SpriteRenderer>().flipX = false;
	Destroy(board);
	StartCoroutine(SetupSceneCoroutine());
	board = GameObject.Find("Board");
 board.SetActive(true);
 Debug.Log("customerselected()-> board.setActive true");
 Player.playerInstance.destinationReached = false;
}
```

Appendix C: Player Script

PC Controls:

```
if (canControl)
{
    if (Input.GetKeyDown("a") && blockX > 0)
    {
        transform.Translate(Vector3.left * movingDistance);
        blockX -= 1;
        GetComponent<SpriteRenderer>().flipX = true;
        LevelManager.LvMg.ProgressBar += stepCount;
    }

    if (Input.GetKeyDown("d") && blockX < 3)
    {
        transform.Translate(Vector3.right * movingDistance);
        blockX += 1;
        GetComponent<SpriteRenderer>().flipX = false;
        LevelManager.LvMg.ProgressBar += stepCount;//move progress bar
    }

    if (Input.GetKeyDown("w") && blockY < 3)
    {
        transform.Translate(Vector3.up * movingDistance);
        blockY += 1;
        LevelManager.LvMg.ProgressBar += stepCount;
    }
    if (Input.GetKeyDown("s") && blockY > 0)
    {
        transform.Translate(Vector3.down * movingDistance);
        blockY -= 1;
        LevelManager.LvMg.ProgressBar += stepCount;
    }
```

Mobile Controls:

```
#region MobileInput
if (Input.touchCount == 1 && canControl) // user is touching the screen with a single touch
{
    Debug.Log("Touched!");
    Touch touch = Input.GetTouch(0); // get the touch
    if (touch.phase == TouchPhase.Began) //check for the first touch
    {
        fp = touch.position;
        lp = touch.position;
    }
    else if (touch.phase == TouchPhase.Moved) // update the last position based on where they moved
    {
        lp = touch.position;
    }
    else if (touch.phase == TouchPhase.Ended) //check if the finger is removed from the screen
    {
        lp = touch.position;  //last touch position. Ommitted if you use list

        //Check if drag distance is greater than 20% of the screen height
        if (Mathf.Abs(lp.x - fp.x) > dragDistance || Mathf.Abs(lp.y - fp.y) > dragDistance)
        {//It's a drag
         //check if the drag is vertical or horizontal
            if (Mathf.Abs(lp.x - fp.x) > Mathf.Abs(lp.y - fp.y))
            {   //If the horizontal movement is greater than the vertical movement...
                if ((lp.x > fp.x) && blockX < 3)  //If the movement was to the right)
                {   //Right swipe
                    transform.Translate(Vector3.right * movingDistance);
                    blockX += 1;
                    GetComponent<SpriteRenderer>().flipX = false;
                    LevelManager.LvMg.ProgressBar += stepCount;//move progress bar
                }
                else if ((lp.x < fp.x) && blockX > 0)
                {   //Left swipe
                    transform.Translate(Vector3.left * movingDistance);
                    blockX -= 1;
                    GetComponent<SpriteRenderer>().flipX = true;
                    LevelManager.LvMg.ProgressBar += stepCount;
                }
            }
            else
            {   //the vertical movement is greater than the horizontal movement
                if (lp.y > fp.y && blockY < 3)  //If the movement was up
                {   //Up swipe
                    transform.Translate(Vector3.up * movingDistance);
                    blockY += 1;
                    LevelManager.LvMg.ProgressBar += stepCount;

                }
                else if (lp.y < fp.y && blockY > 0)
                {   //Down swipe
                    transform.Translate(Vector3.down * movingDistance);
                    blockY -= 1;
                    LevelManager.LvMg.ProgressBar += stepCount;
                }
            }
```

Appendix D: Flash Trigger Script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class FlashTrigger : MonoBehaviour {

       private void OnTriggerEnter(Collider other)
       {
        if (other.tag != "Border")
        other.GetComponent<SpriteRenderer>().color = Color.white;
        if (other.tag == "SurfaceCover")
            Destroy(other.gameObject);
    }
}
```

Appendix E: Level Manager Script

Variables:

```
public class LevelManager : MonoBehaviour {

    public static LevelManager LvMg;
    [Header("basic vars")]
    public int Day;
    public int Money;
    public int ProgressBar;
    public bool GameClear;
    public bool GameOver;

    [Header("game main var's limits")]
    public int DaysLimit = 3;
    public int EarnTarget = 10000;
    public int ProgressBarLimit = 30;

    [Header("linked vars + gameObjs")]
    public int roadMoveSpeed;
    public bool canTalk;
    public bool oneTime = true;
```

```
    //UI parts
    [Header("UI Parts")]
    public GameObject BeforeLvDayDisplay;
    public Text DayDisplay;
    public Text MoneyDisplay;
    public GameObject SelectCMenu;

    //gameObject parts
    [Header("gameObjects parts")]
    public GameObject Taxi;
    public GameObject driverSpawner;
    public GameObject customerSpawner;
    public GameObject driver;
    public GameObject c1_businessman;
    public GameObject c2_femaleStudent;
    public GameObject c3_oldMan;
    public GameObject c4_partTimer;
    public GameObject cSecret_hoodieGirl;

    private bool dayStartRunning;
    private bool dayEnding;
```

Awake(), Update() and Miscellaneous Functions:

```
void Awake () {
    //set tutorial count
    if (!PlayerPrefs.HasKey("TutorialDone") || PlayerPrefs.GetString("TutorialDone") == "false")
    {
        PlayerPrefs.SetString("TutorialDone", "false");
        Debug.Log("tutorialdone = false;");
        gamePaused = true;
    }
    else
        gamePaused = false;
    //PlayerPrefs.SetInt("BadEndCount", 4);
    LvMg = this;
    roadMoveSpeed = 0;
    canTalk = false;
    GameClear = GameOver = false;
    //if cannot reteive saved day count, set a new one
    if (!PlayerPrefs.HasKey("DayCount")) {
        PlayerPrefs.SetInt("DayCount", 1);
        Day = PlayerPrefs.GetInt("DayCount");
    } else {
        Day = PlayerPrefs.GetInt("DayCount");
    }
    //same goes for money earned
    if (!PlayerPrefs.HasKey("MoneyEarned")) {
        PlayerPrefs.SetInt("MoneyEarned", 0);
        Money = 0;
    } else {
        Money = PlayerPrefs.GetInt("MoneyEarned");
    }
    if (PlayerPrefs.HasKey("GameStatus"))
    {
        PlayerPrefs.DeleteKey("GameStatus");
    }
    if (!BeforeLvDayDisplay.activeSelf)
        BeforeLvDayDisplay.SetActive(true);
    Text BeforeDayDisplayText = BeforeLvDayDisplay.GetComponent<Text>();
    BeforeDayDisplayText.text = "Day " + Day.ToString();
    if (PlayerPrefs.GetInt("BadEndCount") != 5)
        StartCoroutine(dayStart());
    BeforeLvDayDisplay.SetActive(false);
    selectiveDisplay = GameObject.Find("SelectiveDisplay");
  }
```

```
void Update () {
 DayDisplay.text = Day.ToString();
 MoneyDisplay.text = "$ " + Money.ToString();

 if (GameClear || GameOver)
 {
     if (GameClear)
         PlayerPrefs.SetString("GameStatus", "GameClear");//for setting game over/clear screen display
     if (GameOver)
         PlayerPrefs.SetString("GameStatus", "GameOver");
     StartCoroutine(FadeLoader.FadeSLoad.Fading(FadeLoader.FadeSLoad.CarRadioScreen));
     Debug.Log("load gameOver Screen");
 }

 if (DialogueManager.DialMg.RideCus01 || DialogueManager.DialMg.RideCus02 ||
     DialogueManager.DialMg.RideCus03 || DialogueManager.DialMg.RideCus04 ||
     DialogueManager.DialMg.RideCus05 || canTalk || dayStartRunning || dayEnding)
 {
     SelectCMenu.SetActive(false);
     //but if customer is selected, have to initiate the enter animation
     if (DialogueManager.DialMg.RideCus01 || DialogueManager.DialMg.RideCus02 ||
         DialogueManager.DialMg.RideCus03 || DialogueManager.DialMg.RideCus04 || DialogueManager.DialMg.RideCus05)
     {
         if (oneTime)
         {
             StartCoroutine(CustomerSelected());
             Debug.Log("customer selected");
             oneTime = false;
         }
     }
 } else {
     SelectCMenu.SetActive(true);
     if (oneTime) //P/S: set to true after every ride
     {
         CustomerVariables.CusVars.CustomersPriceSet();
         oneTime = false;
     }
 }
 if (gamePaused)
     Time.timeScale = 0;
 else
     Time.timeScale = 1;
```

```
public IEnumerator dayStart()
{
    dayStartRunning = true;
    yield return new WaitForSeconds(0.5f);
    //spawn driver & driver enters car
    GameObject Driver = Instantiate(driver, driverSpawner.transform.position, driverSpawner.transform.rotation);
    Animator driverAnim = Driver.GetComponent<Animator>();
    driverAnim.Play("DriverEnterTaxi");
    yield return new WaitForSeconds(driverAnim.GetCurrentAnimatorStateInfo(0).length - SoundManager.soundMg.closeCarDoor_sfx.length);
    SoundManager.soundMg.sfx_source.PlayOneShot(SoundManager.soundMg.closeCarDoor_sfx, 1.0f);
    yield return new WaitForSeconds(SoundManager.soundMg.closeCarDoor_sfx.length);
    Destroy(Driver);
    SoundManager.soundMg.bgm_source.Play();
    dayStartRunning = false;
}
```

```
   public IEnumerator DestinationArrived()
   {
          LevelManager.LvMg.roadMoveSpeed = 5;
          yield return new WaitForSeconds(1.0f);
          LevelManager.LvMg.roadMoveSpeed = 0;
          yield return new WaitForSeconds(0.1f);
          yield return StartCoroutine(LevelManager.LvMg.AfterRideProcess());
          BoardManager.boardManagerInstance.board = GameObject.Find("Board");
          BoardManager.boardManagerInstance.playerP = GameObject.Find("PlayerPrefab(Clone)");
          BoardManager.boardManagerInstance.board.SetActive(false);
          BoardManager.boardManagerInstance.playerP.SetActive(false);
    Debug.Log("destinationarrived()-> board.setActive false");
    selectiveDisplay.SetActive(true);

          oneTime = true;
          Debug.Log("onetime set to TRUE");
          BoardManager.boardManagerInstance.ResetBoard();
   }

   public void AfterRidePayment()
    {
    if (DialogueManager.DialMg.RideCus01) {
        BasicPayment(DialogueManager.DialMg.Cus1TalkCount, CustomerVariables.CusVars.C1_DisplayedPay);
    }
    if (DialogueManager.DialMg.RideCus02) {
        BasicPayment(DialogueManager.DialMg.Cus2TalkCount, CustomerVariables.CusVars.C2_DisplayedPay);
    }
    if (DialogueManager.DialMg.RideCus03) {
        BasicPayment(DialogueManager.DialMg.Cus3TalkCount, CustomerVariables.CusVars.C3_DisplayedPay);
    }
    if (DialogueManager.DialMg.RideCus04) {
        BasicPayment(DialogueManager.DialMg.Cus4TalkCount, CustomerVariables.CusVars.C4_DisplayedPay);
         }
    }

public void BasicPayment(int CurrentCusTalkCount, int PayMoney)
{
    Status_PopUp.statusMg.ReceiveMoney(PayMoney); //then resume with normal payouts
}
```

Appendix F: Dialogue Manager Script

Variables:

```
public class DialogueManager : MonoBehaviour {

    public static DialogueManager DialMg;
    public RPGTalk rpgTalk;

    public bool RideCus01 = false;
    public bool RideCus02 = false;
    public bool RideCus03 = false;
    public bool RideCus04 = false;
    public bool RideCus05 = false;

    public GameObject HoodieGirl_jsImg;

    public bool SkipButtonPressed = false;
    public int Cus1TalkCount, Cus2TalkCount, Cus3TalkCount, Cus4TalkCount;
    private bool InitiateTalk;
```

Awake(), Update() and Miscellaneous Functions:

```
void Awake () {

    //Instantiate(HoodieGirl_jsImg);
    if (!PlayerPrefs.HasKey("Cus1TalkCount") || !PlayerPrefs.HasKey("Cus2TalkCount") ||
        !PlayerPrefs.HasKey("Cus3TalkCount") || !PlayerPrefs.HasKey("Cus4TalkCount"))
    {
        RESET_AllTalkCount();
    }
    if (!PlayerPrefs.HasKey("Cus1BadEnd") || !PlayerPrefs.HasKey("Cus2BadEnd") ||
        !PlayerPrefs.HasKey("Cus3BadEnd") || !PlayerPrefs.HasKey("Cus4BadEnd"))
    {
        RESET_AllBadEnds();
    }

    ResetTempTalkCounts(); //reset the temporary save data on talk counts
    DialMg = this;
    InitiateTalk = true;
}
```

```
void Update () {
        //customer selected
    if (InitiateTalk && LevelManager.LvMg.canTalk)
    {
        if (RideCus01)
        {
            Cus1TalkCount = PlayerPrefs.GetInt("C1TC_Temp"); //initial = 1;

            if (Cus1TalkCount == 5)
            {
                SoundManager.soundMg.bgm_source.Stop();
                rpgTalk.NewTalk("Customer1_Talk5_START", "Cus1Talk5_CutScene1Start", rpgTalk.txtToParse, this, "Cus1Talk5CutScene1");
            }
            else
            {
                rpgTalk.NewTalk("Customer1_Talk" + Cus1TalkCount.ToString() + "_START",
                    "Customer1_Talk" + Cus1TalkCount.ToString() + "_END",
                    rpgTalk.txtToParse, this, "AfterCusTalk");
            }
            InitiateTalk = false;
        }

        if (RideCus02)
        {
            Cus2TalkCount = PlayerPrefs.GetInt("C2TC_Temp"); //initial = 1;

            if (Cus2TalkCount == 5)
            {
                SoundManager.soundMg.bgm_source.Stop();
                rpgTalk.NewTalk("Customer2_Talk5_START", "Cus2Talk5_CutScene1Start", rpgTalk.txtToParse, this, "Cus2Talk5CutScene1");
            }
            else
            {
                rpgTalk.NewTalk("Customer2_Talk" + Cus2TalkCount.ToString() + "_START",
                    "Customer2_Talk" + Cus2TalkCount.ToString() + "_END",
                    rpgTalk.txtToParse, this, "AfterCusTalk");
            }
            InitiateTalk = false;
        }
```

```
public IEnumerator AfterTalk(int CurrentCusTalkCount, string CustomerTalkCount, string CTC_Temp, string badendData, int GoodEndMoney)
{
    //need to show stop car animation first (if not bad end)
    if (CurrentCusTalkCount < 5)
    {
        if (CurrentCusTalkCount == 4)
            Status_PopUp.statusMg.ReceiveMoney(GoodEndMoney);
        CurrentCusTalkCount += 1;//ADD talkcount
        PlayerPrefs.SetInt(CTC_Temp, CurrentCusTalkCount); //temp add
        Debug.Log(CustomerTalkCount + "talkcount +1: " + CurrentCusTalkCount); //display customer talk count
    }

    else if (CurrentCusTalkCount >= 5)
    {
        PlayerPrefs.SetInt(CustomerTalkCount, 1); //reset talkcount
        PlayerPrefs.SetString(badendData, "yes");//save BadEnd data
        PlayerPrefs.SetInt("BadEndCount", PlayerPrefs.GetInt("BadEndCount") + 1);
        Debug.Log(CustomerTalkCount + "BadEnd +1: " + PlayerPrefs.GetInt("BadEndCount")); //display bad ends shown (int)
        LevelManager.LvMg.GameOver = true; //display Game over too
    }
        yield return null;
        InitiateTalk = true;
    LevelManager.LvMg.canTalk = false;
        //Player.playerInstance.canControl = true;
        yield return StartCoroutine(Player.playerInstance.EnablePlayerMove());
        Debug.Log("Can control turn on, after talk(dialogue manager)");
  }
```

```
public void AfterCusTalk()
{
    if (RideCus01) {
        StartCoroutine(AfterTalk(Cus1TalkCount, "Cus1TalkCount", "C1TC_Temp", "Cus1BadEnd", CustomerVariables.CusVars.Customer1GoodEndPay));
    }
    else if (RideCus02) {
        StartCoroutine(AfterTalk(Cus2TalkCount, "Cus2TalkCount", "C2TC_Temp", "Cus2BadEnd", CustomerVariables.CusVars.Customer2GoodEndPay));
    }
    else if (RideCus03) {
        StartCoroutine(AfterTalk(Cus3TalkCount, "Cus3TalkCount", "C3TC_Temp", "Cus3BadEnd", CustomerVariables.CusVars.Customer3GoodEndPay));
    }
    else if (RideCus04) {
        StartCoroutine(AfterTalk(Cus4TalkCount, "Cus4TalkCount", "C4TC_Temp", "Cus4BadEnd", CustomerVariables.CusVars.Customer4GoodEndPay));
    }
    else
        return;
}
```

Appendix G: *itch.io* Website of **I'm The Skilled Driver**

I'm The Skilled Driver (3 Day Cab)

We're **2ez Studios** who made I'm The Skilled Driver(3 Day Cab), a story-driven game where a taxi driver does his usual job - driving customers here and there.

Go on with his usual routine in the **3 days** and get that sweet, sweet $**10000** cash, and that's it.
Yeah, that simple. (Ok, perhaps not that simple now that we've added a bit of puzzle element in it to make this less boring...)

We probably didn't follow the theme, now that I said the game's objective..... (cries in corner)

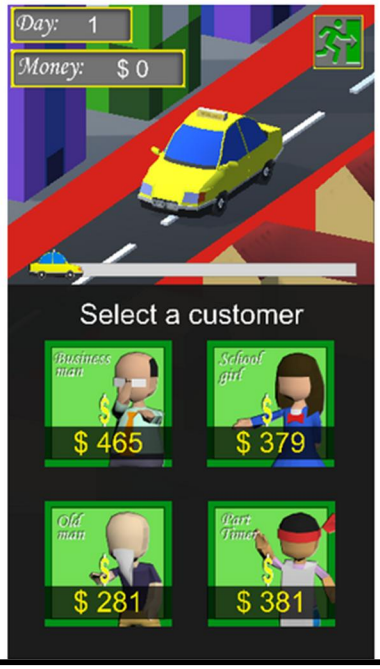
But hey, we did pour in effort into doing this, so enjoy playing!
(this is how you play hard to get, right?)

[P/S: Due to lack of creativity in naming, we decided to change the name from '3DayCab' to 'I'm The Skilled Driver'. We apologize for any inconvenience caused.]

How to Play:
- Just use your mouse/finger. That's pretty much what you need anyways.

More information

Published	21 days ago
Status	Prototype
Platforms	Windows, Android
Author	Hegback
Genre	Visual Novel
Tags	Horror, Ludum Dare 42
Links	Ludum Dare



Download

Download	[PC]3DayCab.zip (v0.1)	23 MB
Download	3DayCab.apk (v0.1)	32 MB
Download	[PC]TSDriver.zip (v0.2)	25 MB
Download	ImTheSkilledDriver (v0.2)	36 MB