

Sweet REVENGE

Design Document
February 20, 2009
Proposed by Team Pink

The work submitted in this document is solely our own and the Honor Code was neither bent nor broken.

Jillian Cornette _____

Christa Jones _____

Yangyang Liu _____

Nerina Martinez _____

Table of Contents

Overview	1
Concept Artwork	
Main Characters	2
Coco	
Spork King	
Minor Characters	
Casey	3
Slushy	3
Gooley	3
Chips	3
Mayor Marzipan	4
Loll E. Pop	4
Redd Hot	4
Bon Bon Bay	5
Coco's House	6
City Hall	7
Factories	8
Fluff Factory	
Gummy Grocery	
Chocolate Corner	
Hardcandy Company	9
Sweet Treet	10
Marshmallow Meadow	11
Gummy Bear Forest	12
Chocolate Swamp	13
Level Puzzles	
Slushy	14
Gooley	15
Chips	16
Weapons & Health	17
Script	18
Storyboard	27
Technical Design	
Software	35
User Case Diagram	35
Flow Chart	36
Software Architecture	37
Class Descriptions	38
Class Relationships	41
Class Breakdown	42
Risk Mitigation Plan	54

Overview

Welcome to Bon Bon Bay! This delightful hub of activity is home to a variety of sweet confections including our faithful cupcake hero, Coco. The story begins when Coco discovers the Spork King and his Army of Utensils are brainwashing all the candy of Bon Bon Bay and have kidnapped Casey, Coco's wife. The chaos that the army spreads begins to turn all candy against one another. Coco must explore the lands of Bon Bon Bay and help the other candies in order to find clues as to where Casey is being held.

On his quest to save Casey, Coco encounters several other colorful locals who provide him with clues and helpful suggestions. He also finds help at Bon Bon Bay's boutiques and shops including Fluff Factory, Gummy Grocery, and Chocolate Corner. Only after collecting the destroyed enemies from the surrounding lands and retrieving the key from the special characters within each land, can the village shops be opened. The enemy pieces can only be restored in their appropriate shops and in return Coco will receive a clue to Casey's whereabouts.

Don't be fooled by Marshmallow Meadow's sweet, wintry appearance. While the puffy, pink bunnies that hop through this land may appear innocent at first glance, they have been corrupted by the evil Spork King's Utensil Army, and they aren't afraid to kick! Coco's most effective weapon against these enemies is the microwave gun.

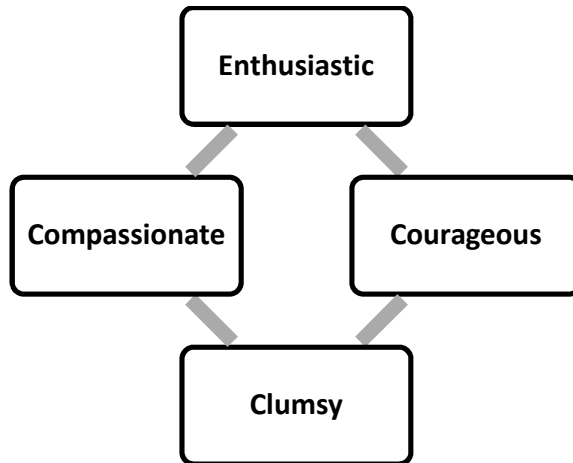
While exploring Marshmallow Meadow, Coco must find Slushy the Snowman who has lost all of his items (hat, arms, buttons, nose, etc.) and needs Coco's help! After finding all of these items amongst a winter candy scene, he will give Coco the key to the Fluff Factory.

Colorful, angry gummy bears corrupted by the Army of Utensils inhabit the wooded area of Gummy Bear Forest. Beware, they bite! Coco should use the flame gun to destroy them. Coco will also encounter the gummy owl Gooey, who needs help feeding her babies. She is prideful, however, and challenges Coco to collect more gummy worms than her before dinner time. Only when Coco has successfully completed Gooey's challenge will he receive the key to the Gummy Grocery. Once Coco has conquered the gummy bears and retrieved the key, he may return the enemy pieces to Gummy Grocery.

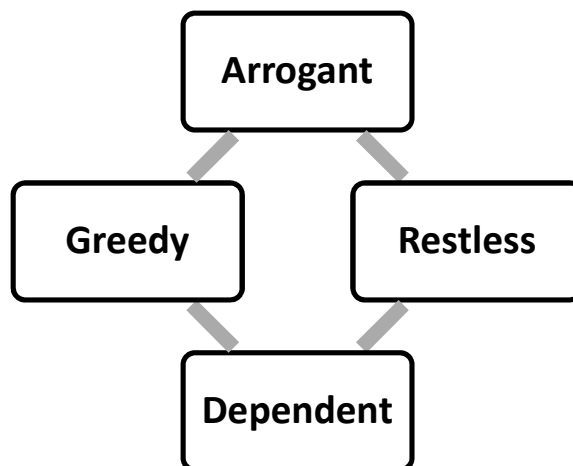
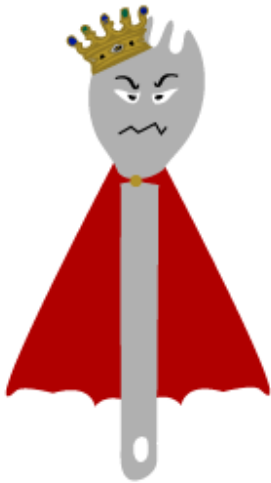
In the murky waters of this balmy, Chocolate Swamp lie chocolate alligators turned vicious by the Utensil Army. They now think cupcakes are a delicious snack! The freeze gun should be used for the most effective results in destroying these gators. Chips, the chocolate turtle lives here and is in need of some serious help. His shell is in disarray and he won't give Coco the key to the Chocolate Corner until his shell is restored to its pristine state. Coco must defeat all of the chocolate alligators and return them to Chocolate Corner.

Main Characters

Coco



Spork King



Minor Characters

Casey



Special Land Characters

Slushy



Gooley

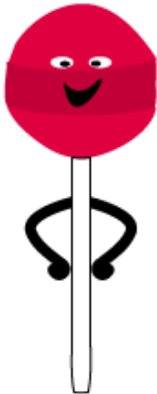


Chips



**Mayor Marzipan**

The Mayor of Bon Bon Bay is always in a rush and seems to be a little suspicious.

**Loll E. Pop**

This helpful treat is always up to date about the gossip around Bon Bon Bay. She'll keep you in the know.

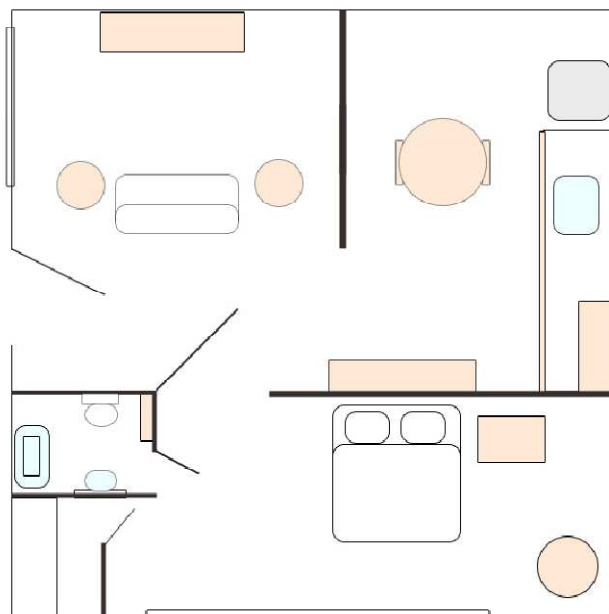
**Redd Hot**

This firecracker owns Hardcandy Company and doesn't mind showing off his merchandise.

Coco's House

Mood: Coco's house is a bright yellow oven. The mood is calm and peaceful.

Acoustic: It will feature a more subtle variation of the tropical sounds of Bon Bon Bay. For the intro cinematic, when Coco opens the pantry the sound abruptly changes to one that is daunting.



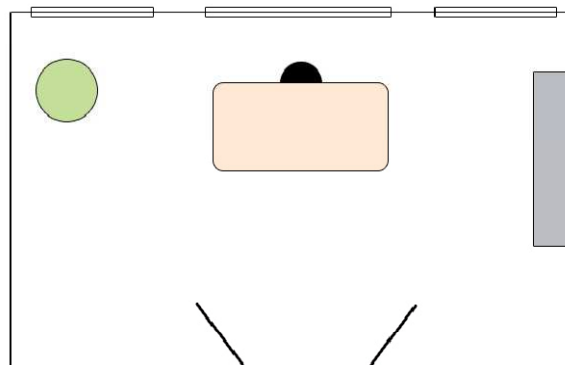
City Hall

Mood: Formal and more serious

Acoustic: The music will not be as bright and lighthearted as other places in Bon Bon Bay. It would be slower, with a more formal tone that matches with the government building.



Mayor Marzipan Office Floor Plan



Factories

Fluff Factory

Mood: Peaceful and upbeat

Acoustic: A mixture of Bon Bon Bay music with subtle accents of Marshmallow Meadow acoustics.



Gummy Grocery

Mood: Peaceful and upbeat

Acoustic: A mixture of Bon Bon Bay music with subtle accents of Gummy Bear Forest acoustics.



Chocolate Corner

Mood: Peaceful and upbeat

Acoustic: A mixture of Bon Bon Bay music with subtle accents of Chocolate Swamp acoustics.



Hardcandy Company

Mood: Calm, western

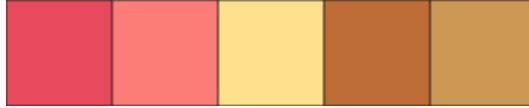
Acoustic: More of a country tone to the background music to match the character of the owner.



Sweet Treats

Mood: Upbeat and joyful

Acoustic: Similar to that of Bon Bon Bay with certain subtle higher pitch tones



Marshmallow Meadow

Mood: Seemingly serene; however there is a sense of apprehension in the air.

Acoustic: Mostly light and airy with some hints of suspense.



Level Map:



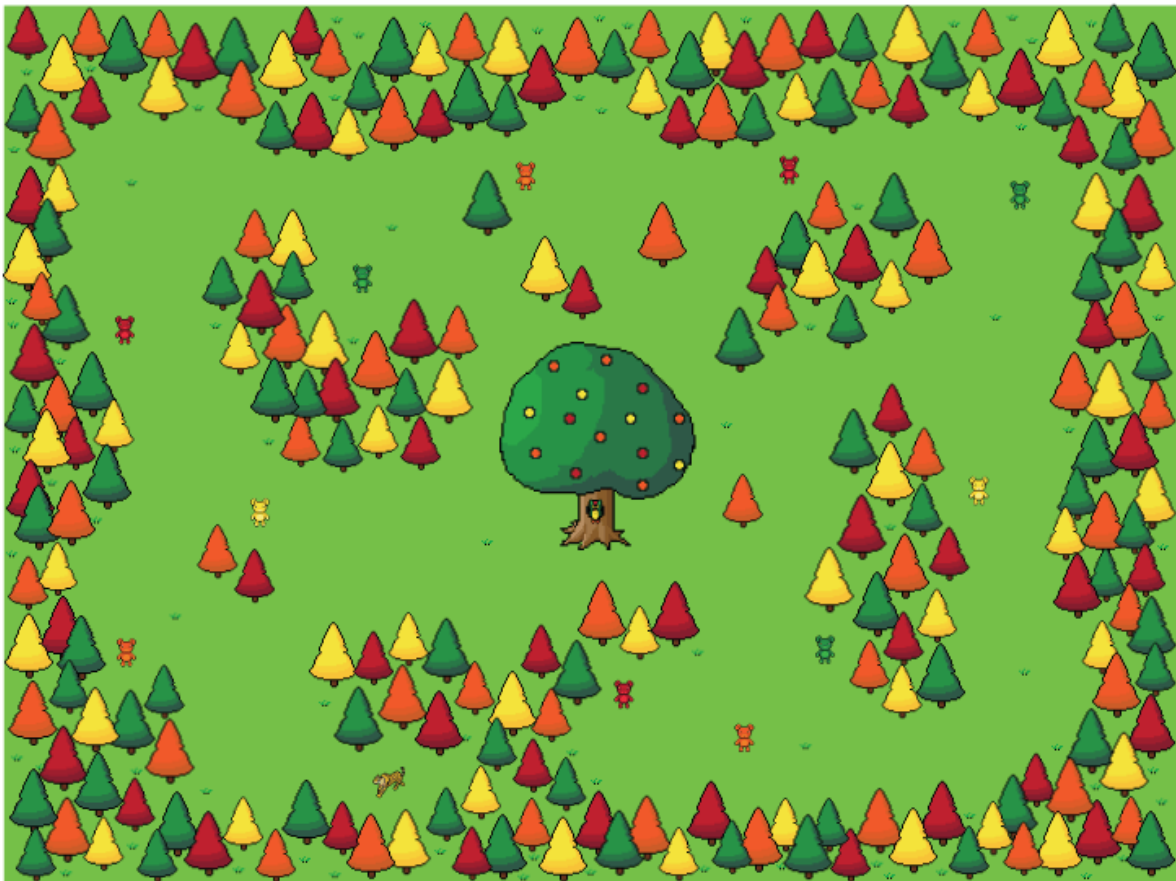
Gummy Bear Forest

Mood: The bright colors contrast the evil enemies; cautious

Acoustic: The background noise will fall somewhere in between that of Marshmallow Meadow and Chocolate Swamp; not too light and airy, but not too dark and eerie. Sound effects we will need are an owl hoot, bear growl and a tiger roar.



Level Map:



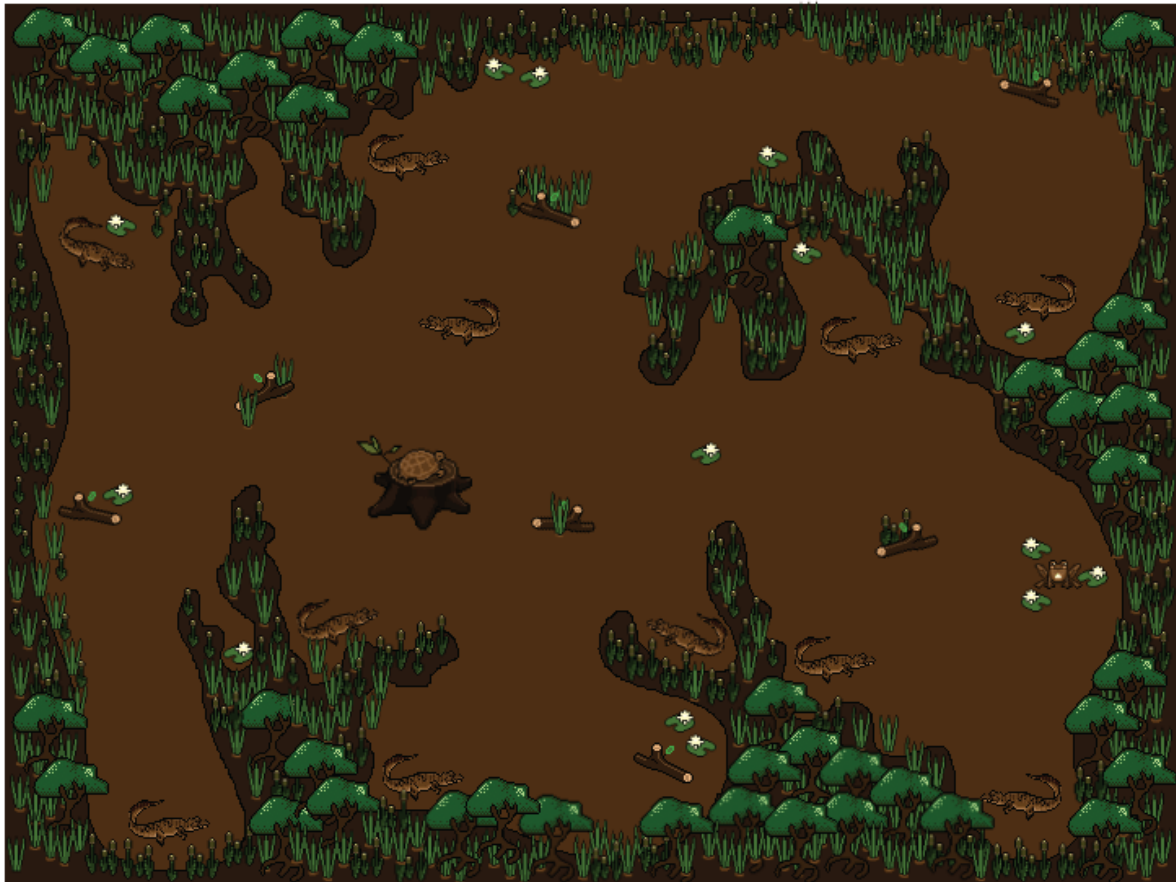
Chocolate Swamp

Mood: It does not seem as peaceful as the other lands; dark and eerie

Acoustic: Background music almost like a suspenseful film. Sound effects we'll need are alligator chomps and frog "ribbit" sounds.



Level Map:

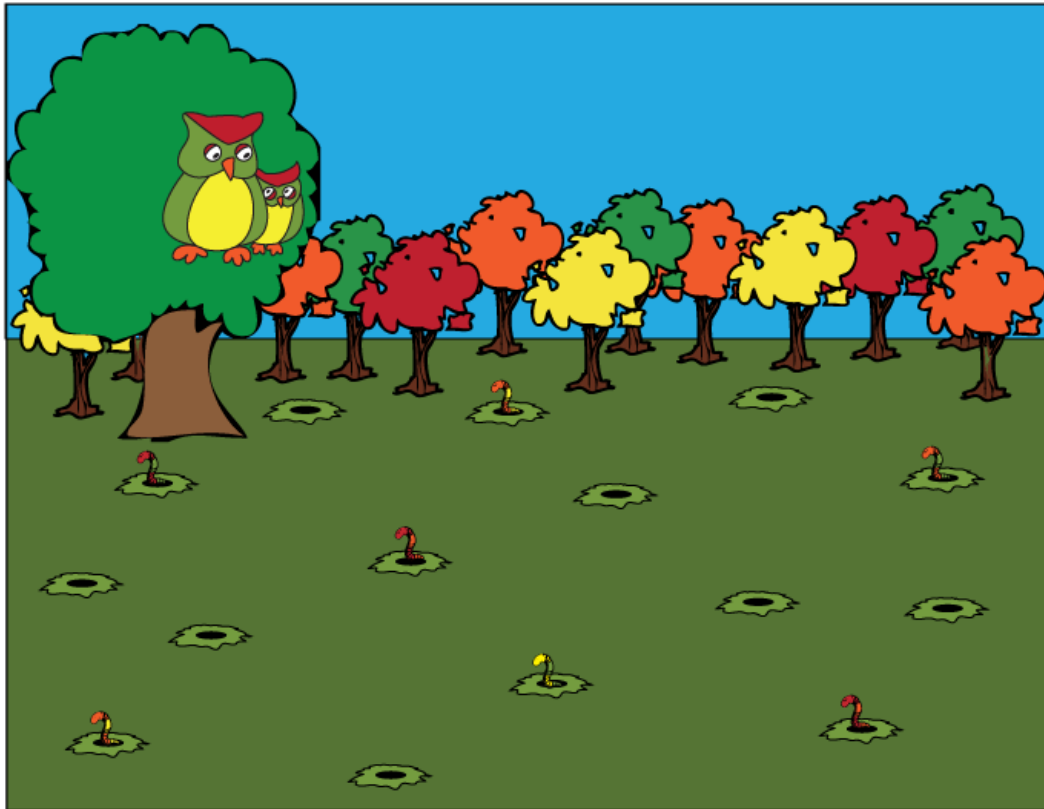


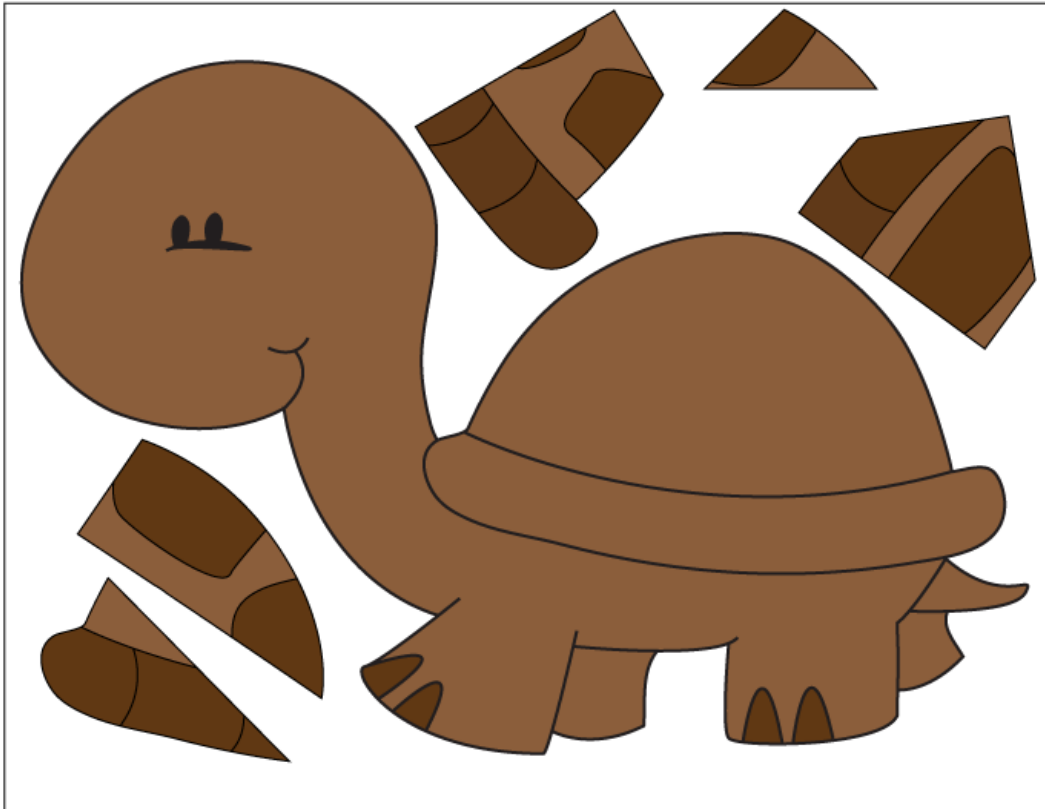
Puzzles

Marshmallow Meadow – Find Slushy's Parts



Gummy Bear Forest – Collect the Gummy Worms



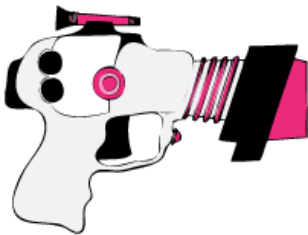
Chocolate Swamp – Put Chips' Shell Back Together

Weapons and Health

Range is dependent on distance between Coco and enemy

Farthest	25% from 0-5
	50% from 0-10
	75% from 0-15
Closest	100% from 0-20

Weapons not used in their respective lands are much less effective



Microwave Gun

Shoots microwaves (looks like sine waves) and causes bunnies to expand and explode into pieces.



Flame Gun

Shoots out fire and melts the gummy bears into puddles



Freeze Gun

Shoots out frost/snowflakes and freezes the chocolate alligators



Health Pack

These health packs will be scattered throughout Bon Bon Bay and the surrounding lands. Coco takes a gamble when picking them up as they can help or hurt him.

Script

Character Voice Descriptions

- Coco Male Chipper, not too deep
- Spork King Male Deep raspy voice
- Loll E. Pop Female Younger “valley-girl” voice
- Mayor Marzipan Male Middle age
- Redd Hot Male Deep western/cowboy accent
- Slushy Male Young
- Gooey Female Motherly, calm voice
- Chips Male Old, slow grandpa voice
- Bunnies Female Young, hyper
- Gummy Bears Male Young, squeaky
- Alligators Male Young, deeper
- Elf_1 Male Snappy, flippant
- Elf_2 Female Kind, caring, sweet
- Elf_3 Male Laid back, Australian accent

Bon Bon Bay - morning

The scene opens with an aerial shot of Bon Bon Bay. A noise that sounds like a timer constantly going off can be heard. The scene zooms into Coco’s home, an oven, and through the window until the alarm clock can be seen on the nightstand. It is blinking and continuing to beep. A hand reaches out from under the covers and turns it off but knocks it off the nightstand in the process. Coco gets out of bed and pulls his silver wrapper over his patterned “boxers” wrapper.

COCO

Hmm...(yawns)... breakfast ...(scratches)

Coco walks to the kitchen -

COCO

Casey! You gonna make breakfast?
(no response) Cereal it is...

Coco walks to the pantry and opens it. On the inside of the door hangs a note pinned with a spork. It reads -

COCO

You shall never see your delicious Casey again!
King Spork.

SPORK KING

(voice over

Mwahahahaha!

COCO

Noooo! I must find Mayor Marzipan, he'll know
what to do.

*When Coco finds Mayor Marzipan in his office, he is reading from
a file that says "TOP SECRET" -*

COCO

Mayor! Mayor! Help! The Spork King has taken Ca--

MAYOR

Not now Coco...The town is in shambles! That damn
Spork and his Utensil Army. They're turning all
the candy citizens against one another. How could
this happen? I never thought I would have to
resort to this... (he looks down at the file)

VOICE

Mayor hurry! You've got to see this!

*The mayor drops the file on his desk and rushes out of his
office.*

COCO

I wonder what he was reading...

*Coco finds the file on the desk. It contains Sketches/blueprints
for three different guns (microwave, freeze, flame).*

COCO

Why would the mayor have these?

*Coco takes the file with the sketches/blueprints. Upon further
investigation, the sketches/blueprints show a header stating
"The Hardcandy Company. 105 Gumdrop Place. Bon Bon Village"*

COCO

Maybe they will know more about these.

Once Coco leaves the Mayor's office, Loll E. Pop approaches Coco.

LOLL E. POP

Coco, I heard about Casey.

COCO

Word travels fast here.

LOLL E. POP

Word on the street is The Utensil Army moved through the three lands last night. I don't know much but maybe some of the citizens from Marshmallow Meadow, Gummy Bear Forest, or Chocolate Swamp can help you. Be careful, rumor is that the Spork King has cursed some of the candies. From what I've heard, they may not be as friendly as their old selves.

COCO

Thanks! I'll try my luck there.

Once Coco finds the Hardcandy Company building –

REDD HOT

Howdy. What can I do ya' for?

COCO

Have you heard what the Spork King has done?

REDD HOT

Well I reckon I have. Things are getting pretty awful out there. But I'd put my money on it being pretty cotton pickin' safe in here. (points to his guns hanging on the wall)

COCO

The Spork King has kidnapped my wife, Casey, and I must find her.

REDD HOT

I s'pose I could loan you one of these here guns. I got some mighty fine looking guns...

COCO

I can see that, but I think I might need something a little more specific...(hands Candy the blueprints)

REDD HOT

Hmmm... I ain't never seen nothing like this before. Ya' know I could get in some big trouble for this. So it's gonna cost you big time.

COCO

Like?

REDD HOT

I reckon somewhere around 100 gumdrops.

COCO

What?!? I don't have that kind of money!

REDD HOT

Well I heard that the ice cream store across town is hiring. Maybe you could pick up some extra cash there.

If Coco tries to enter any of factories before finding the key – Flash to sign on the door that reads: "Sorry we're closed."

COCO

I wish I had a key. I wonder where I could get one.

When Coco meets the Marshmallow Snowman, Slushy –

SLUSHY

Hi there Coco. I'm having such a terrible day! Can you please help me find my pieces, I seem to have misplaced them during all the mayhem. I could let you borrow the key to the Fluff Factory. There you may drop off the injured bunnies so that they may restored to their sweet selves.

Cut to - Snowman game screen - display rules - "Find and return all of Slushy missing pieces!"

When Coco meets the Gummy Owl, Gooley –

GOOEY

Coco, is that you? I need your help. I'm trying to feed my youngins, but I can't seem to find enough food. Will you please help me collect worms in time for dinner? If you help me, I might be able to part with my spare key to Gummy Grocery.

Cut to - Owl game screen - display rules - "Collect more worms than Gooley before time runs out!"

When Coco meets the Chocolate Turtle, Chips –

CHIPS

Coco, thank goodness you're here! The Utensil Army ran over my shell and shattered it to pieces! Can you please put it back together for me? All I have is the key to the Chocolate Store to repay you for your efforts.

Cut scene - Turtle puzzle screen - display rules - "Help Chips by putting his shell back together!"

*If Coco returns to the Chocolate Corner with a key
Cut to - Inside Factory*

ELF_1

Hey, how did you get in here?

COCO

Ummm...(holds up the key)

ELF_1

Alright then, give me that. How can I help you?

COCO

(hands him the key) Well...I was hoping maybe you could help them...(puts the bag with chocolate pieces on the counter clumsily)

If Coco hasn't killed any enemies yet -

ELF_1

(looks in the bag) Is this some kind of practical joke? It's empty! Come back when you've got some work for me!

If Coco has killed at least one enemy -

ELF_1

(looks in the bag, gasps) Oh my! Okay! But it won't be easy.

COCO

Just let me know when they're back to normal so I can ask them some questions. Thanks!

If Coco returns to the factory before the elf has notified him -

ELF_1

I've still got a lot of work to do! I'll let you know as soon as I'm finished!

If Coco returns to the Gummy Grocery with a key -

ELF_2

Where did you get a key to my store?

COCO

I did Gooley a favor and he let me borrow it. Do you think you could give it back to her for me?

ELF_2

Sure, no problem! What else can I help you out with?

COCO

I thought maybe you could help me turn these bunnies back to their friendly selves.

ELF_2

Let me take a look. (looks in the bag)

If Coco hasn't killed any enemies yet -

ELF_2

Oh. Seems like you might have made a mistake. I think this bag is empty. Are you sure you needed my help?

If Coco has killed at least one enemy -

ELF_2

Hmmm... This doesn't look too good, but I think I can help.

COCO

I would really appreciate it. Could you please let me know as soon as they are back to normal?

ELF_2

Will do!

If Coco returns to the factory before the elf has notified him-

ELF_2

I'm not quite finished yet. Do you think you could come back a little later?

*If Coco returns to the Fluff Factory with a key
Cut scene - Inside Factory*

ELF_3

Whoa, dude, where did you come from?

COCO

Ummm...outside? I helped Slushy out and he gave me this key.

ELF_3

That's pretty awesome bro.

COCO

I know, right? So do you think you could help me turn some bunnies back to normal?

ELF_3

I guess it wouldn't hurt for me to take a look.
(looks in the bag)

If Coco hasn't killed any enemies yet -

ELF_3

Bummer, man. This bag is empty.

If Coco has killed at least one enemy -

ELF_3

Totally gnarly dude. I'll try to get them back to their rad selves as soon as possible.

COCO

Sweet. Could you let me know when you get done so I could talk them about what happened?

ELF_3

Definitely man.

If Coco returns to the factory before the elf has notified him-

ELF_3

Sorry bro, I'm not done yet. Try back a little later on.

Once the first enemies have been restored (bunnies, gummy bears, or alligators) -

Cut scene - inside of respective factory

ENEMY_1

What a bad dream...

COCO

I'm so glad you are feeling better. But I was wondering if I could ask you a few questions.

ENEMY_1

Sure, but I'm afraid my memory has been a bit fuzzy.

COCO

Well just try to remember as much as you can. Any details will help.

Once the second enemies have been restored (bunnies, gummy bears, or alligators) -

Cut scene - inside of respective factory

ENEMY_2

How in the world did I get here?

COCO

Ummm....It's kinda a long story. Perhaps I could ask you a few questions?

ENEMY_2

Of course. I'm not sure if I'll be much help though.

Once the third enemies have been restored (bunnies, gummy bears, or alligators) -

Cut scene - inside of respective factory

ENEMY_3

Ugh. My head is killing me!

COCO

Yeah...Sorry about that.

ENEMY_3

Sorry? For what?

COCO

Oh nevermind. Do you mind if I ask you a few questions?

ENEMY_3

Yeah, go ahead. Shoot

COCO

(under his breath)
I've been doing a little too much of that lately.

ENEMY_3

What?

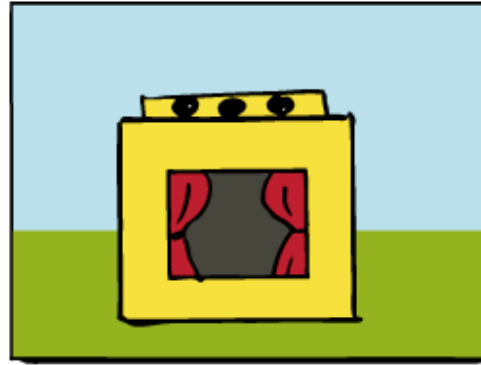
COCO

Oh nothing...

Storyboard



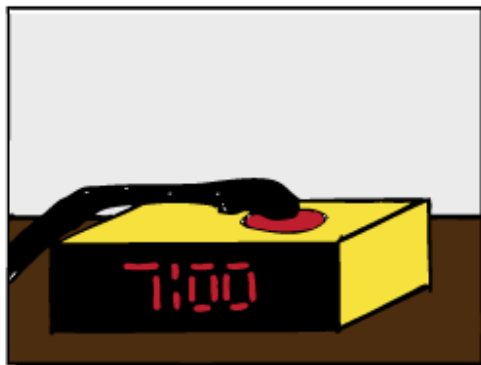
Aerial shot: Bon Bon Bay
Alarm beep sounds. **Zoom in** on Coco's house.



Level Camera Angle: Continue **zoom in**
through window. Alarm beep continues.



Zoom: We see Coco's bed and nightstand.
Beeping continues. Coco begins moving.



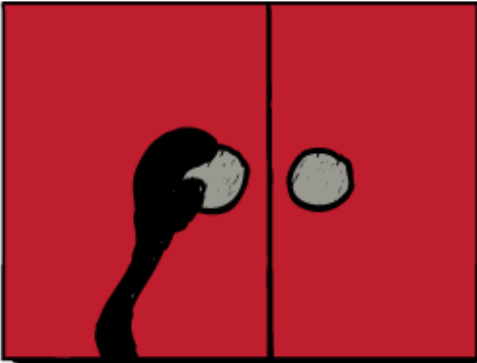
Jump Cut: Coco hits the button, beeping stops.



High Hat: Coco gets out of bed and yawns putting on his wrapper. The clock falls to the floor.



Cut to kitchen. Over the shoulder: Camera pans left and right as Coco calls for Casey.
Static shot as Coco walks to pantry.



Close up: Coco opens pantry...



Cut to: Note left by the Spork King



Cut to: Coco reacting to the note.



Cut to: City Hall Static shot



Cut to: Inside the Mayor's office. Coco calls out to the mayor who is facing the opposite direction.



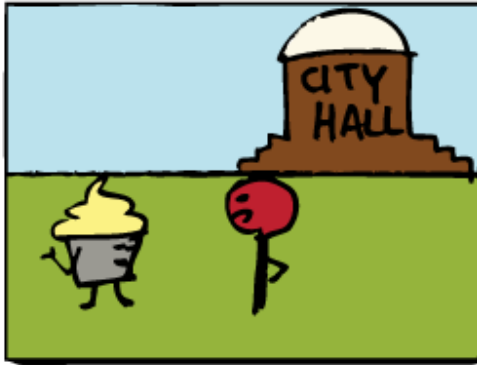
Over the shoulder: Mayor is looking through a file and seems distracted as Coco talks.



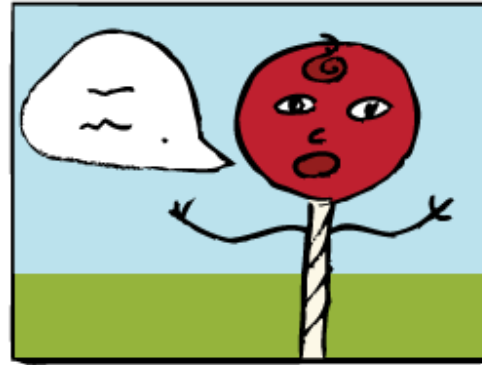
Mayor Marzipan is called from another room, drops file on desk and runs out of the scene. Coco turns around and sees the file on the desk.



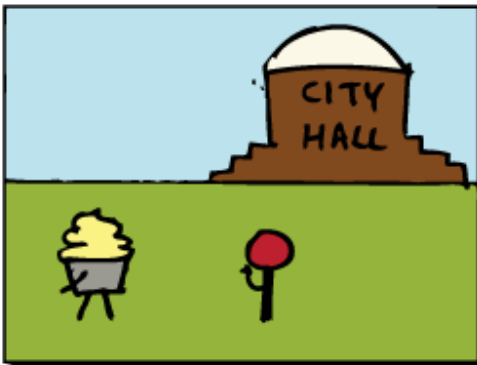
POV: Coco holding the file.



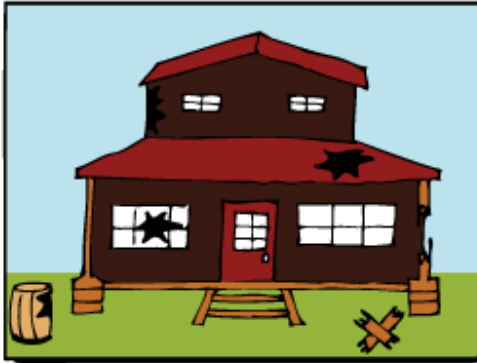
Cut to: Outside Coco is talking to Loll E. Pop.
City Hall is seen in the background



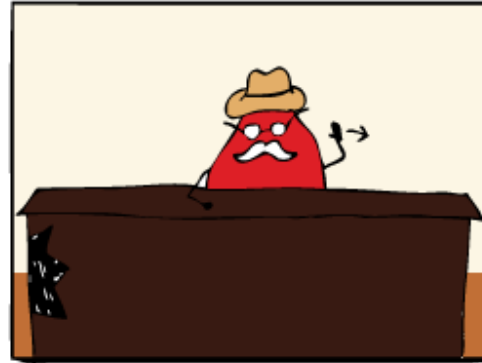
Close up: Loll E. Pop is talking



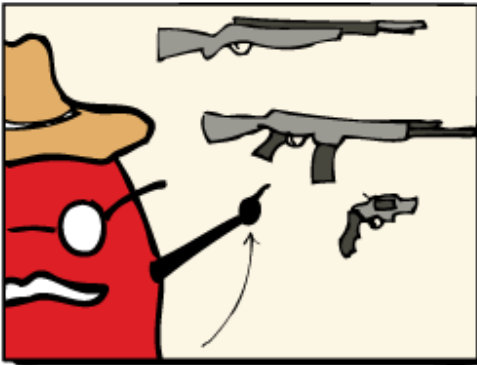
Cut to intro shot. Coco walks out of the scene.



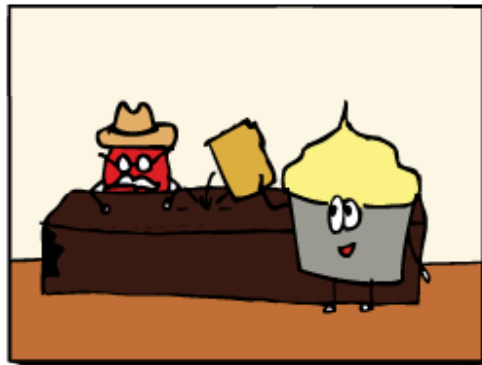
Cut to: Hardecandy Company



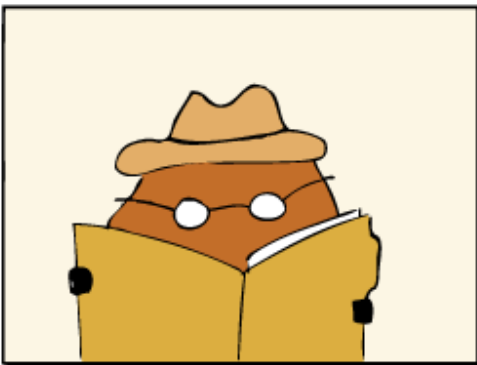
POV: Redd Hot waves as Coco walks in.



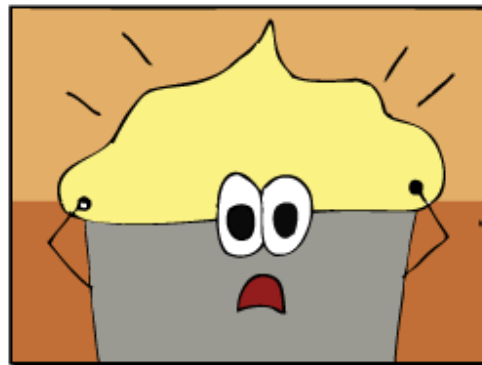
Cut to: Close up on Redd Hot points to guns on the back wall.



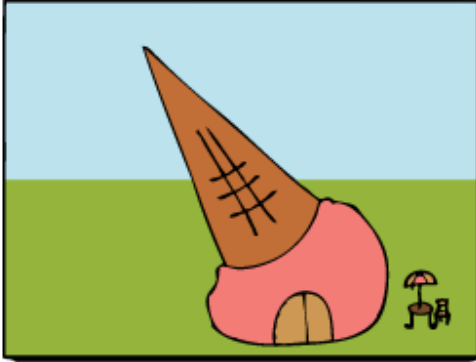
Coco hands the file to Redd Hot.



Cut to: Close up on Redd Hott looking through file.



Cut to: Close up on Coco reacting to Redd Hott telling him the guns will cost 100 gumdrops.



When Coco walks up to ice cream shop,
Cut to shot of Sweet Treats.



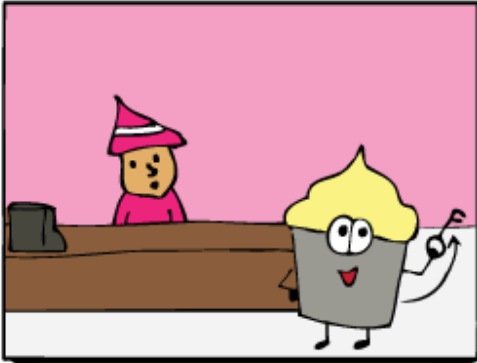
Cut to: Coco's **POV** inside Sweet Treats at counter with Elf.



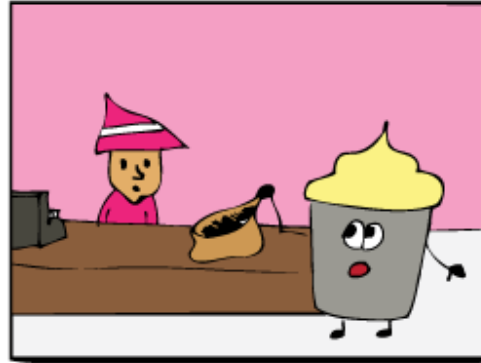
Coco walks up to the factory and tries to enter without a key.



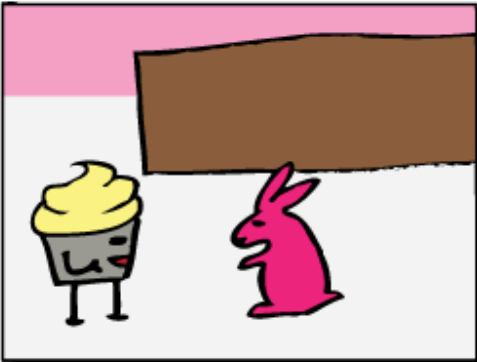
Close up: Sign



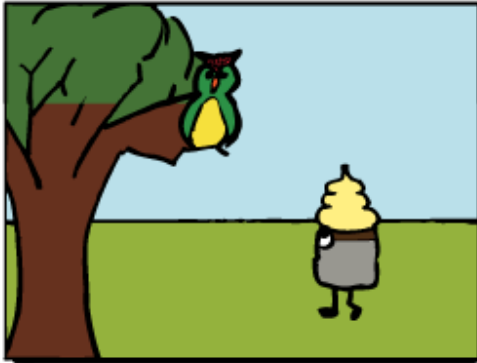
Inside a factory, Elf is behind the counter. When asked how he got in, Coco holds up the key.



Inside a factory, Coco puts bag on the counter.



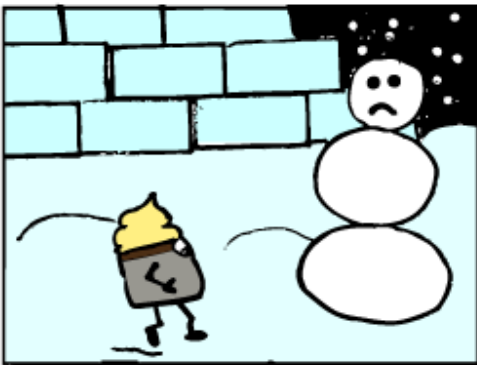
Inside a factory with a restored enemy. Coco and the enemy are in front of the counter talking.



Coco walks up to Gooley in the Gummy Bear Forest.



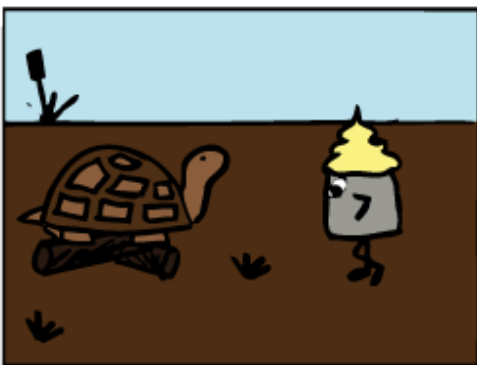
Close up on Gooley while she's talking.



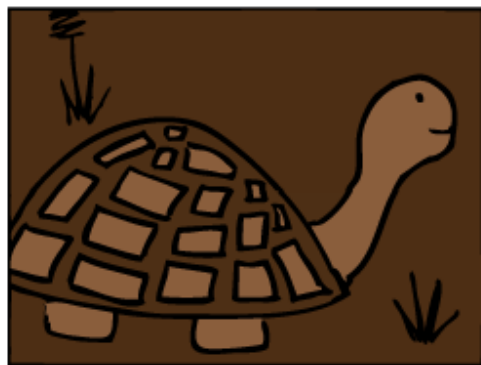
Inside the igloo in Marshmallow Meadow, Coco walks up to Slushy.



Close up on Slushy while he's talking.



Coco walks up to Chips in the Chocolate Swamp.



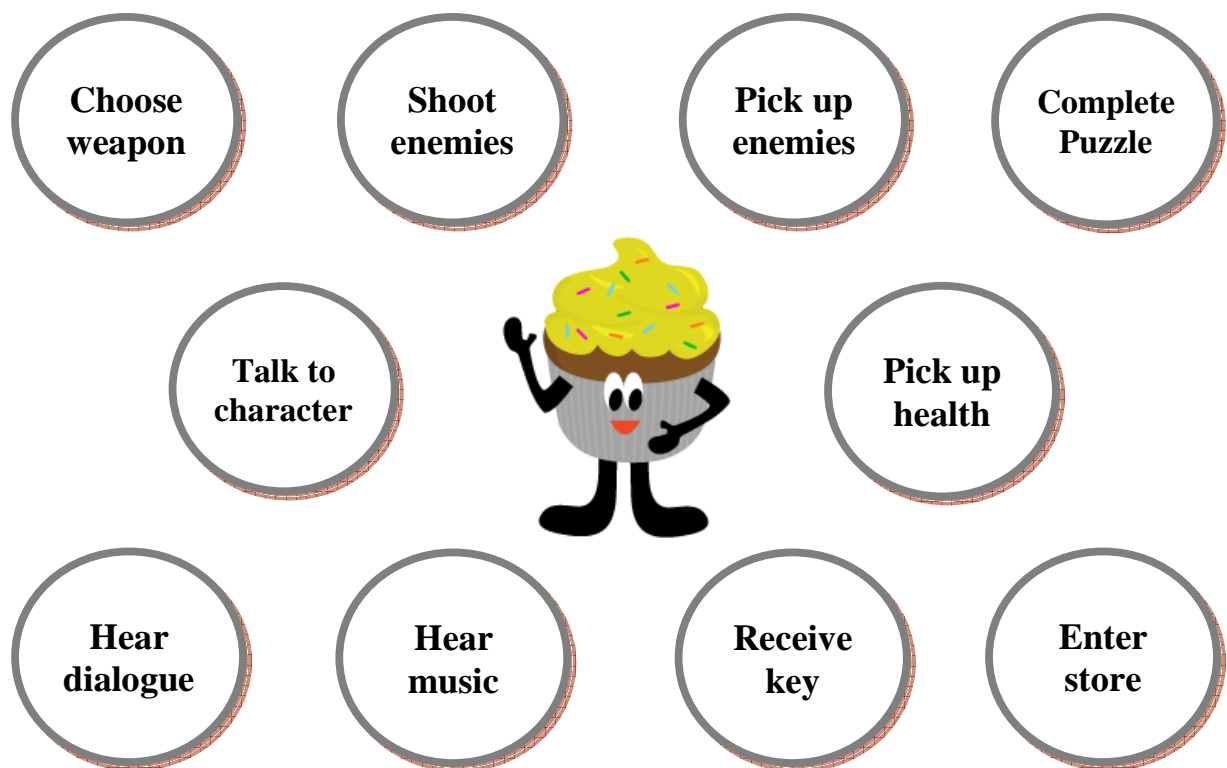
Close up on Chips while he's talking.

Technical

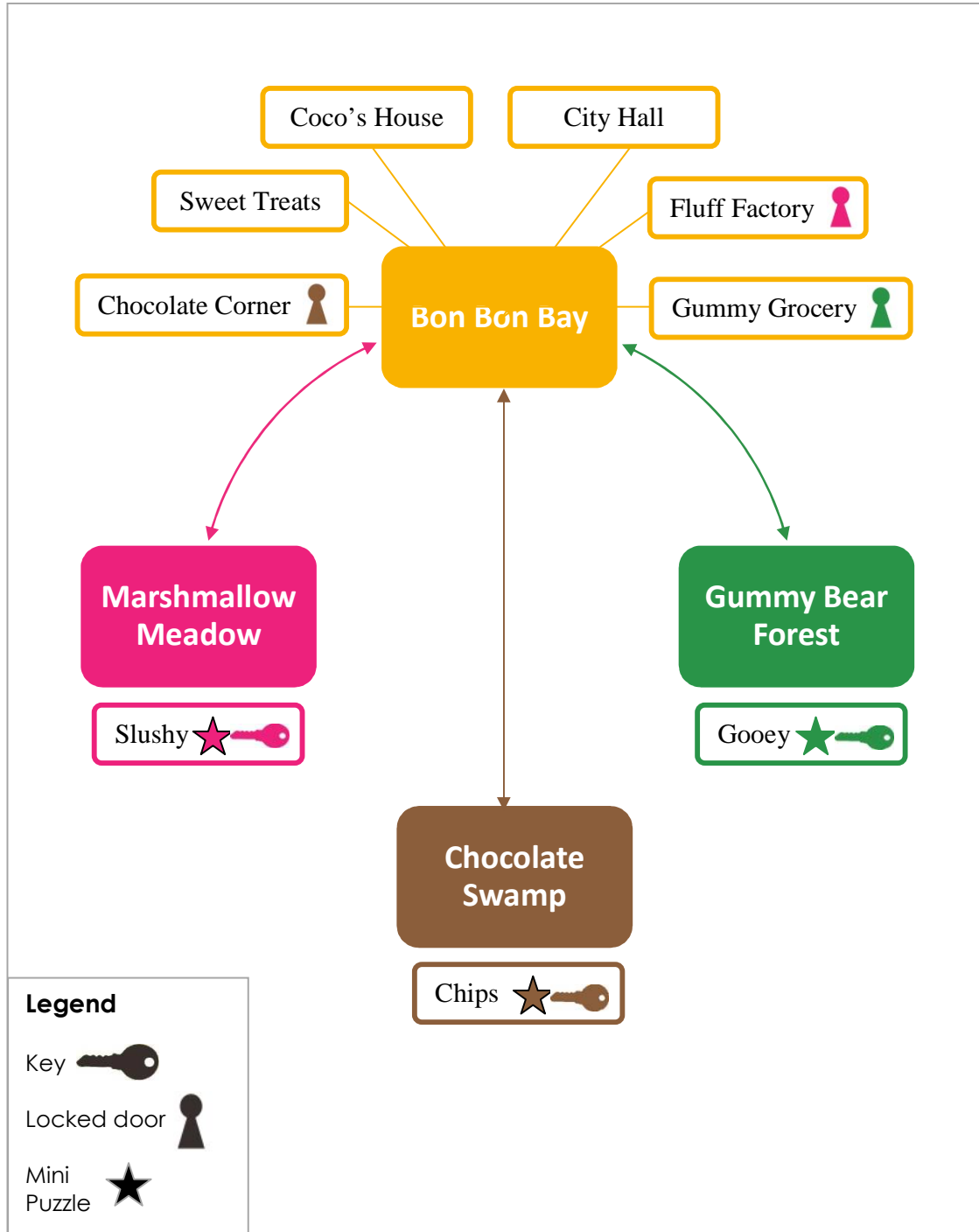
Software

We will implement this game using Java. The graphics will be loaded as .png images to allow for transparency. We will be using Fruity Loops to create sound and music and JFugue to import the music as midi files.

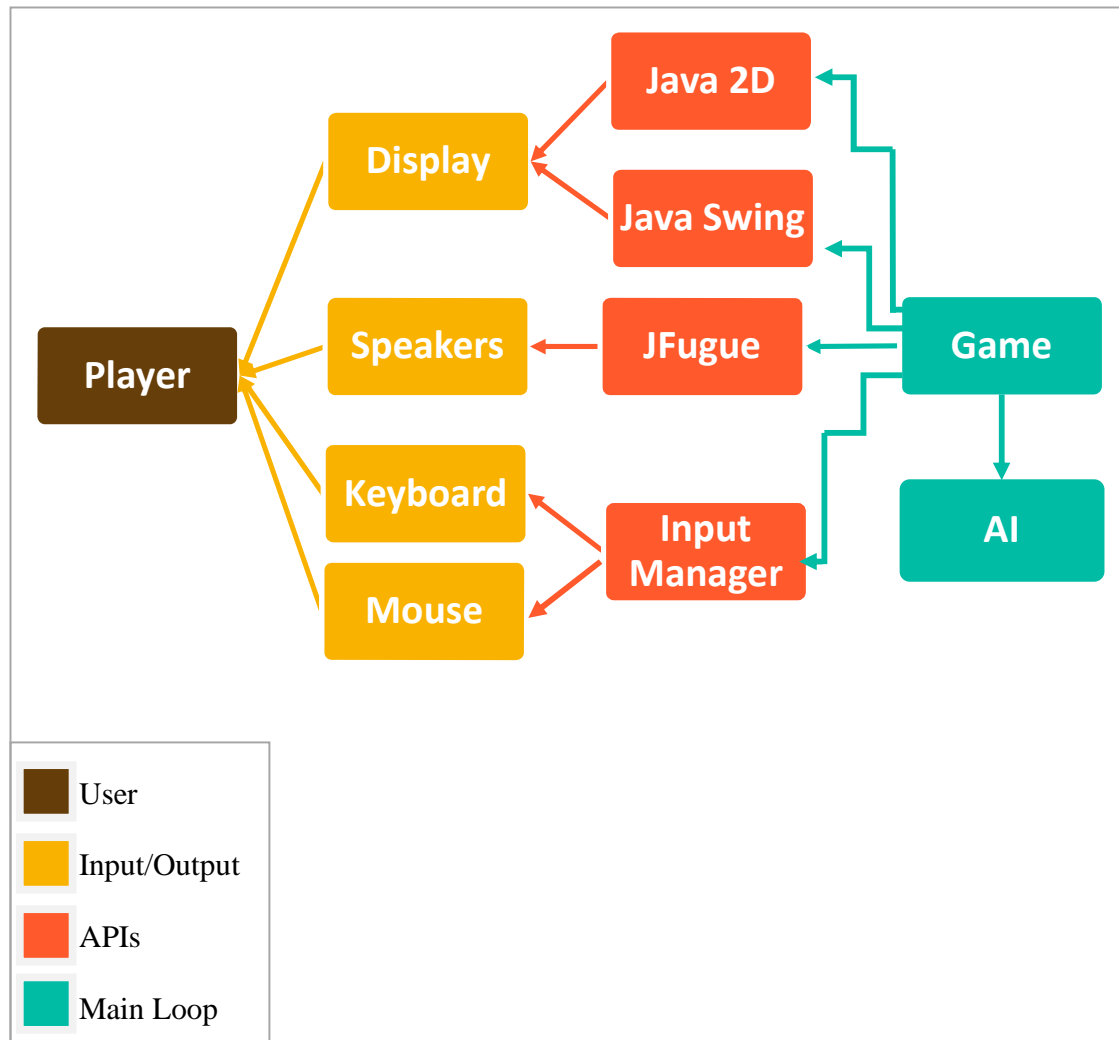
User Case Diagram



Flowchart



Software Architecture



Class Descriptions

AmbientSound
-soundFile: file
+AmbientSound(File file)
+play(): void

Bear
+Bear(File file, BufferedImage image)
+move(): void

Coco
-position: Position
-hitPoints: int
-damage: int
-dead: boolean
-close: boolean
-weapon: Weapon[]
-dialogue: String[]
-image: BufferedImage
+Coco(File file, BufferedImage image)
+move(int direction): void
+takeDamage(int damage): void
+shoot(): void
+getPosition(): Position
-setPosition(Position p): void
+getHitPoints(): int
-setHitPoints(int hitPoints): void
+isDead(): boolean
+heal(int hitPoints): void
+speak(): String
-calculateRange(Enemy e): double

Enemy
-position: Position
-hitPoints: int
-damage: int
-dead: boolean
-close: boolean
-enemyImage: BufferedImage
+Enemy()
+move(): void
+takeDamage(int damage): void
+attack(): void
+getPosition(): Position
-setPosition(Position p): void
+getHitPoints(): int
-setHitPoints(int hitPoints): void
+isDead(): boolean
+closeEnoughToBite(): boolean
+getImage(): BufferedImage
-setImage(BufferedImage image): void

Bunny
+Bunny(File file, BufferedImage image)
+move(): void

ChipsGame
+ChipsGame(File file, BufferedImage image)
-displayInstructions(): void
-play(): void

CookieJar
-p: Position
-item: Item
-image: BufferedImage
-open: boolean
+CookieJar(File file, BufferedImage image)
+openJar(): boolean
+setOpen(boolean open): void

CutScene
-movieImages[]: BufferedImage
+CutScene(BufferedImage image[])
+play(): void

DialogueCharacter
-position: Position
-close: boolean
-dialogue: String[]
-image: BufferedImage
+DialogueCharacter(File file, BufferedImage image)
+getPosition(): Position
-setPosition(Position p): void
+inRange(): boolean
+speak(): String
+getImage(): BufferedImage
-setImage(BufferedImage image): void

GameFrame
+GameFrame()
-display(): void
-createGameObjects(): void
-restart(): void

Gator
+Gator(File file, BufferedImage image)
+move(): void

GunFire
-gunFireImage: BufferedImage[]
-position: Position
+GunFire(BufferedImage gunFireImage[], Position p)
+getGunFireImage(): BufferedImage
-setGunFireImage(BufferedImage gImage[])
+getPosition(): Position p
-setPosition(Position p): void

Map
-music: Music
-sound: Sound[]
-specialCharacter: SpecialCharacter
-dialogueCharacter: DialogueCharacter[]
-mapImage: BufferedImage
+Map(File mapFile, BufferedImage image)
+getCookieJars(): CookieJar[]
-setCookieJars(CookieJar cj[]): void
+getMusic(): Music
-setMusic(Music music): void
+getSound(): Sound[]
-setSound(Sound sound[]): void
+getSpecialCharacter(): SpecialCharacter
-setSpecialCharacter(SpecialCharacter sc): void
+getDialogueCharacter(): DialogueCharacter[]
-setDialogueCharacter(DialogueCharacter dc[]): void
+getMapImage(): BufferedImage
-setMapImage(BufferedImage mapImage): void
+drawMap(): void

SlushyGame
+SlushyGame(File file, BufferedImage image)
-displayInstructions(): void
-play(): void

SpecialCharacter
-position: Position
-close: boolean
-dialogue: String[]
-hasKey: boolean
-characterImage: BufferedImage
+SpecialCharacter(File file, BufferedImage image)
+getPosition(): Position
-setPosition(Position p): void
+speak(): String
+hasKey(): boolean
+inRange(Position p): boolean
+getImage(): BufferedImage
-setImage(BufferedImage image): void

GoopyGame
+GoopyGame(File file, BufferedImage image)
-displayInstructions(): void
-play(): void

Item
-itemImage: BufferedImage
+Item(File file, BufferedImage image)
+use(): void

MiniGame
win: boolean
+MiniGame()
-displayInstructions(): void
-play(): void
+missionsAccomplished(): boolean

Music
-musicFiles[]: File
+Music(File file)
+play(File musicFile): void
+stop(): void

PlayerActivatedSound
-soundFile: file
+PlayerActivatedSound(File file)
+play(): void

Position
-x: int
-y: int
-direction: int
+Position(int x, int y, int direction)
+getX(): int
+setX(int x): void
+getY(): int
+setY(int y): void
+getDirection(): int
+setDirection(int direction): void
+screenToMap(Position p): Position
+mapToScreen(Position p): Position

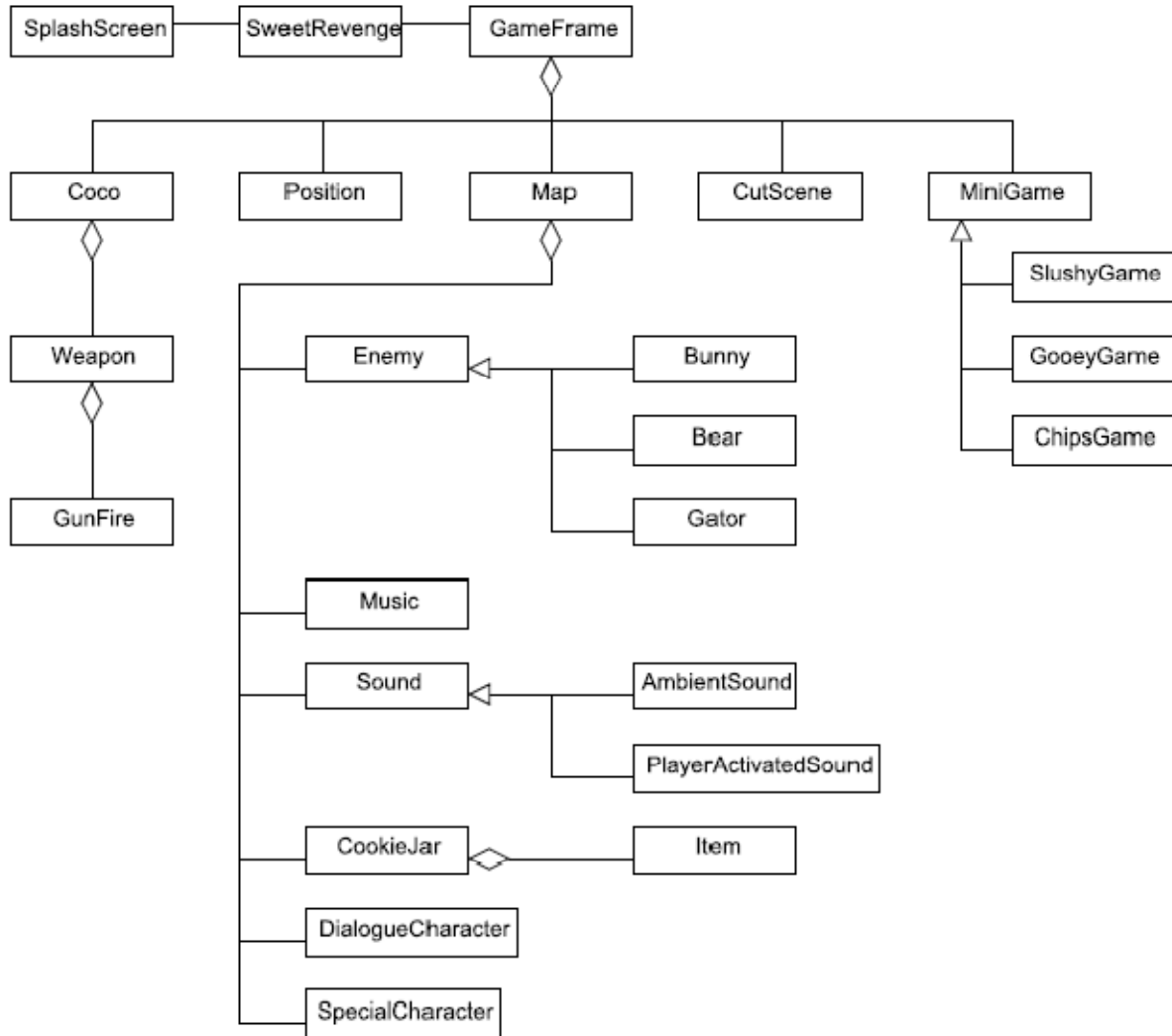
SplashScreen
-play: boolean
+SplashScreen()
-play(): boolean

SweetRevenge
+main()

Weapon
-damage: int -inUse: boolean -gunFire: GunFire
+Weapon(boolean inUse, GunFire gf)
+shoot(): void +inUse(): boolean +getGunFire(): GunFire -setGunFire(GunFire gf): void

Sound
-soundFile: File
+Sound()
+play(): void

Class Relationship Diagram



Class Breakdown

Class SweetRevenge

The main class; calls the SplashScreen class first as the introductory screen to the game, when the user selects the play option from the SplashScreen, the main class will call the GameFrame class which will display the actual game window

Class SplashScreen

Displays the introductory screen to the game; will contain the player options to play or exit; upon the user's selection to play, the splash screen will close and return a boolean to the main class to create the game window

Constructor Summary

SplashScreen()
creates a new SplashScreen

Method Summary

boolean play()
exits the splash screen and signals to start/exit the game

Class GameFrame

Displays the game window; will contain the player options to restart the game or exit; upon the user's selection to restart the game, all game objects will be reinitialized and the user will start from the beginning; creates and initializes all game objects

Constructor Summary

GameFrame()
creates a new GameFrame

Method Summary

void display()
displays the map, characters, and all other visual game objects and scenery

void createGameObjects()
initializes all game properties and objects

void restart()
reinitializes all game properties and objects to restart the game

Class Map

Contains information on an individual level map, including: map image, special characters, dialogue characters, cookie jars, level music, ambient sounds, player activated sounds, etc; creates all level-specific objects

Constructor Summary

Map(File mapFile, BufferedImage mapImage)
creates a new Map using the data from the specified map file and image

Method Summary

void drawMap()
draws the map and all game level objects to the screen

CookieJar[] getCookieJars()
returns an array of cookie jars on this level map

DialogueCharacter[] getDialogueCharacters()
returns an array of dialogue characters on this level map

BufferedImage getImage()
returns the map image

Music getMusic()
returns the level music

Sound[] getSounds()
returns an array of level sounds

SpecialCharacter[] getSpecialCharacters()
returns an array of special characters on this level map

void setCookieJars(CookieJar cj[])
sets the level's cookie jars to the cookie jars in the given array

void setDialogueCharacters(DialogueCharacter dc[])
sets the level's dialogue characters to the dialogue characters in the given array

void setImage(BufferedImage image)
sets the map image to the given image

void setMusic(Music m)
sets the level music to the given music

void setSounds(Sound s[])
sets the level's sounds to the sounds in the given array

void setSpecialCharacters(SpecialCharacter sc[])
sets the level's special characters to the special characters in the given array

Class Position

Stores the map x- and y-coordinates of an object and provides the ability to convert from map coordinates to screen coordinates as well as screen coordinates to map coordinates; also stores the direction that the object is facing (north, northeast, east, southeast, south, southwest, west, and northwest)

Constructor Summary

Position(int x, int y, int direction)
creates a new Position at the map coordinates (x, y) in the specified direction

Method Summary

int getDirection()
returns the direction of the object

int getX()
returns the x coordinate of the object

int getY()
returns the y coordinate of the object

Position mapToScreen(Position p)
returns a Position object with screen coordinates calculated from map

coordinates

Position screenToMap(Position p)

returns a Position object with map coordinates calculated from screen coordinates

void setDirection(int direction)

sets the direction of the object to the specified direction

void setX(int x)

sets the x coordinate of the object to the specified x-value

void setY(int y)

sets the y coordinate of the object to the specified y-value

Class Coco

The class for our main character; stores information specific to Coco (hit points, image, weapons carried, position, dialogue, etc); allows Coco to speak, move, attack, take damage, and heal

Constructor Summary

Coco(File cocoFile, BufferedImage cocoImage)

creates a new Coco using the information in the given file and the given image

Method Summary

void shoot()

checks if Coco has a weapon in hand and calls that weapon's shooting methods

double calculateRange(Enemy e)

calculates and returns the range from Coco to the specified enemy

int getHitPoints()

returns Coco's current number of hit points

int getPosition()

returns Coco's position

void heal(int hitPoints)

increments Coco's hit points by the specified amount, but does not

allow Coco to have more than the maximum amount

boolean isDead()

returns true if Coco's hit points are less than 1

void move(int direction)

updates Coco's position by moving him one space in the given direction

void setHitPoints(int hitPoints)

sets Coco's hit points to the number specified

void setPosition(Position p)

sets Coco's position with the specified position

String speak()

returns a string of Coco's dialogue

void takeDamage(int damage)

decrements Coco's hitpoints by the specified damage number

Class MiniGame

An abstract class; contains an abstract play method to be implemented by its subclasses; tracks if the minigame was completed and the player accomplished the task

Constructor Summary

MiniGame()

creates a new MiniGame

Method Summary

void displayInstructions()

abstract method that displays the minigame instructions at the beginning of the game

void play()

abstract method that specifies minigame play

boolean missionsAccomplished()

returns true if the player accomplished the task

Class SlushyGame

Extends the MiniGame class; specifies functionality and gameplay for Slushy's minigame (a search game that asks the user to click the screen image to help find Slushy's pieces)

Constructor Summary

SlushyGame(File minigameFile, BufferedImage minigameImage)
creates a new SlushyGame with the given minigame image

Method Summary

void displayInstructions()
displays the minigame's instructions

void play()
tracks the player's progress in the minigame and provides its functionality

Class GooleyGame

Extends the MiniGame class; specifies functionality and gameplay for Gooley's minigame (a timetrial that asks the player to collect a number of gummy worms before Gooley does)

Constructor Summary

GooleyGame(File minigameFile, BufferedImage minigameImage)
creates a new GooleyGame with the given minigame image

Method Summary

void displayInstructions()
displays the minigame's instructions

void play()
tracks the player's progress in the minigame and provides its functionality

Class ChipsGame

Extends the MiniGame class; specifies functionality and gameplay for Chips' minigame (a puzzle game that asks the player to reassemble Chips' chocolate turtle shell)

Constructor Summary

ChipsGame(File minigameFile, BufferedImage minigameImage)
creates a new ChipsGame with the given minigame image

Method Summary

void displayInstructions()
displays the minigame's instructions

void play()
tracks the player's progress in the minigame and provides its functionality

Class CutScene

Stores the still images that will comprise each cut scene; plays the cut scenes

Constructor Summary

CutScene(BufferedImage sceneImages[])
creates a new CutScene with the specified array of images

Method Summary

void play()
displays a sequence of images

Class Weapon

Specifies what type of weapon is currently in use and the image and damage for that weapon

Constructor Summary

Weapon(boolean inUse, GunFire gunFire)

Sets the gun fire to be displayed for the respective weapon, and sets the boolean true if we are displaying this particular gun

Method Summary

void shoot()

Is called when the user clicks to shoot, and calls the gunFire image and animation

boolean inUse()

Returns true if the gun is the current one in use, false otherwise

GunFire getGunFire()

Returns the GunFire object that is currently being displayed

void setGunFire(GunFire gunFire)

Set the gun fire image and position

Class GunFire

Each weapon has a different type of gunfire; the gunfire tracks its position in order to detect whether or not an enemy has been hit

Constructor Summary

GunFire(BufferedImage gunFire, Position p)

image is set to one of the following microWaves.png, fireBalls.png, frost.png and the map x,y coordinates and the direction (Moore's Neighborhood) of the gunfire

Method Summary

BufferedImage getGunFireImage()

returns the image of gun fire

void setGunFireImage(BufferedImage bflmage)

set the gun fire image to the bflmage

Position getPosition()

returns the position object of the gun fire

```
void setPosition(Position p)
    set the position (x, y, direction) of the gun fire based on Coco's position
```

Class SpecialCharacter

Contains the character information and position of a stationary special character in each of the 3 enemy lands that Coco must talk to and complete a task for in order to retrieve a key to open up a shop in the mainland

Constructor Summary

```
SpecialCharacter(File characterFile, BufferedImage characterImage)
    creates a special character with the specified image and file
    information
```

Method Summary

```
Position getPosition()
    returns the position object of special character
```

```
void setPosition(Position p)
    set the position (x, y, direction) of the special character
```

```
String speak()
    returns the string of dialogue for the character to speak
```

```
boolean hasKey()
    returns true if the special character still has his/her factory key
```

```
boolean inRange(Position p)
    returns true if the special character is in range of Coco, and thus can
    begin to talk
```

```
BufferedImage getImage()
    returns the special character's current image
```

```
void setImage(BufferedImage void)
    sets the current image of the special character
```

Class Music

Responsible for playing the background music during different stages of the game.

Constructor Summary

Music(File file)
Get the music file to be played.

Method Summary

void play(File musicFile)
Play the music file.

void stop()
Stop the music.

Class Sound

The super class that is responsible for the sound effect of the game.

Constructor Summary

Sound()
Get the sound file to be played.

Method Summary

void play()
Play the soundFile.

Class AmbientSound

Extends Sound class, is responsible to play AmbientSound in the background.

Constructor Summary

AmbientSound(File file)
Get the sound file to be played.

Method Summary

void play()
Play the sound file.

Class `PlayerActivatedSound`

Extends `Sound` class, is triggered when the player takes action.

Constructor Summary

`PlayerActivatedSound(File file)`
Get the sound file to be played.

Method Summary

`void play()`
Play the sound file.

Class `Enemy`

An abstract class; contains the information relevant to this enemy, including: hit points, damage, position, image, etc; provides functionality for enemy movement, attacks, damage received, etc

Constructor Summary

`Enemy()`
creates a new `Enemy`

Method Summary

`void attack()`
damages `Coco` with the enemy's hit point damage

`boolean closeEnoughToBite(Position p)`
returns true if the enemy is close enough to `Coco` to bite

`int getHitPoints()`
returns the enemy's number of hit points

`BufferedImage getImage()`
returns the enemy's image

`Position getPosition()`
returns the enemy's position

`boolean isDead()`

returns true if the enemy is dead

void move()

updates the enemy's position by calculating a new path

void setHitPoints(int hitPoints)

sets the enemy's hit points to the specified number of hit points

void setImage(BufferedImage image)

sets the enemy's image to the specified image

void setPosition(Position p)

sets the enemy's position to the specified position

void takeDamage(int damage)

decrements the enemy's hit points by the specified damage number

Class Bunny

Extends the Enemy class; contains a specific algorithm for choosing a path

Constructor Summary

Bunny(File bunnyFile, BufferedImage bunnyImage)

creates a new Bunny with the specified file and image

Method Summary

void move()

chooses a path, moves, and updates the position of the Bunny

Class Bear

Extends the Enemy class; contains a specific algorithm for choosing a path

Constructor Summary

Bear(File bearFile, BufferedImage bearImage)

creates a new Bear with the specified file and image

Method Summary

void move()

chooses a path, moves, and updates the position of the Bear

Class Gator

Extends the Enemy class; contains a specific algorithm for choosing a path

Constructor Summary

Gator(File gatorFile, BufferedImage gatorImage)
creates a new Gator with the specified file and image

Method Summary

void move()
chooses a path, moves, and updates the position of the Gator

Class DialogueCharacter

Contains the information for a specific stationary dialogue character; information logged includes the character's position, dialogue, and image; contains functionality to check if the character and Coco are in close proximity

Constructor Summary

DialogueCharacter(File file, BufferedImage image)
creates a new DialogueCharacter by using the given file's information and image

Method Summary

getPosition()
returns the position of the character

setPosition()
sets the position of the character

boolean inRange(Position p)
returns true if Coco is within the DialogueCharacter's range to talk

String speak()
returns the dialogues of the character as a String

BufferedImage getImage()
returns the image of the character

void setImage()
sets the image for the specific character

Class CookieJar

Contains the position and item that the cookie jar is hiding; cookie jars are like gift boxes that are spread out over the map that contain special items that will help Coco on his journey

Constructor Summary

CookieJar(File file, BufferedImage cookieJarImage)
creates a new CookieJar with the information specified in the file and the image

Method Summary

boolean openJar()
returns true if the cookie jar is open

void setOpen(boolean open)
if open is true - sets the cookieJarImage to and open cookie jar, otherwise set the image to the closed cookie jar

Class Item

a special, helpful item that benefits Coco on his journey (can heal Coco's hit points)

Constructor Summary

Item(File file, BufferedImage itemImage)
sets the image of the item

Method Summary

void use()
this method is called when an item is picked up, it will update the display of the item

Mitigation Plan

We may run into situations in which we may not be able to implement every aspect of the game as we intended due to time constraints or other technical or artistic issues. This plan takes these issues into account and explains what will be done under the following circumstances.

Case 1 - Dialogue

We would like to import character dialogue as audio files. This requires actors and recording equipment. If the means by which we will execute this plan do not seem realistic, then the dialogue will appear on the screen with character name labels. In the worst case, the dialogue will print out to the command prompt.

Case 2 - Graphics

If we are unable to import the graphics, all characters will be represented as Java 2D objects. Coco will be represented as a yellow square. The enemies will be red circles, and all other special characters as blue triangles.

Case 3 - Audio

If we are unable to import the audio files, we will compose the music in JFugue which requires no use of midi files, just the use of JFugue API. In the worst case scenario, there will be no music in the game and the dialogue will be read as text.

Case 4 - Cut Scenes

In the case in which we are unable to implement the cut scenes, a screen will pop up with the script and a sill picture of the characters involved in that particular scene. If we are unable to import the images,

Case 5 - Shooting

“Shoot and scatter” is the animation which occurs when Coco shoots and kills an enemy. The enemy is scattered into several pieces and must be picked up by Coco. If we are unable to implement the “shoot and scatter” animation, Coco will automatically obtain this enemy in his possession and the enemy will simply disappear from the screen.

Case 6 - Puzzles

In the case that the mini puzzles cannot be implemented into the game, the user will automatically receive the key to the corresponding shop.

Case 7 - Emergency

In the event of illness/emergency, the other group members will distribute the additional responsibilities amongst themselves.

Case 8 – Cookie Jar

We would like the health packs to be packs of sprinkles and Coco will sprinkle them on his frosting and gain health. If we are unable to implement this, the health packs will be

cookie jars; Coco will walk up to the cookie jar and will gain health points.

Case 8 – Time Constraints

If we are unable to finish the entire game, there are certain aspects of the game that must be implemented and others that are not required. Bon Bon Bay will be the first land we implement. The required functions of this land are Coco's house and the three shops. The implementation of the inside of the shops is not entirely required.

Next, we will implement the lands based on difficulty; therefore we will begin with Marshmallow Meadow. Here we must have the enemies. The snowman, Slushy is not a requirement as we can give Coco the key when he kills all the enemies if necessary. The implementation of the remaining lands is desired but not required.