

Design Studio 3

Informatics 121

Party Haus

The Collaborative Dollhouse

Design Team:

Tianran Zhang

Nelson Hsu

Preston Trieu

Priyanka Saba

Timothy Bington

Table of Contents

1. Project Introduction.....	2
2. Design Methods.....	3
3. Application Design.....	4
4. Interaction Design.....	6
5. Architecture Design.....	11
6. Implementation Design.....	12
7. Log.....	20
8. Artifacts.....	23

Project Introduction

The name, Party Haus, comes from the German word for House. We want our company to go above the boundaries of language and culture in this more interconnected world. In fact, very little of the interaction between the user and our interface will necessitate the use of a Western keyboard. We instead use emojis. This will allow our users to interact without the barrier of language or if they do not have an in-depth knowledge of reading and writing.

The projected audience, those who would get to use our project, will be kids from ages four to seven, both boys and girls. Their parents will also be interacting with the software, albeit at a lower level. They will look over the kid's progress, and make sure that their kid is safe. The organization behind Party Haus prioritizes kid safety and seeks to work cooperatively with the parents to ensure their safety.

The projected stakeholders, who we need to satisfy in the development of the product, are toy companies and investors, gaming control board (gcb), and parents. We need the product to appeal to toy companies and investors in order to understand how to design our product. We have to make it safe and playable so we aren't violating any laws that the gcb oversees and so that the parents will be at ease knowing their children will not leak any important or sensitive information online.

Our goals center around the core of collaborative play. We want to create the best experience for our audience, kids and parents alike. Although the primary goal given was to create a dollhouse, we do not want to make a dollhouse, at least by title, because that is a toy generally associated with the female gender. Our goal is, therefore, to make a game that both genders will enjoy. We want our interface to be user-friendly; it should be easy to play the game and understand how to play the game. The game should be able to be saved so that the kids do not have to start over every time, and the kids should be able to access their software from anywhere if they have the login and password combo and the application installed. We want to keep the kids engaged so that they will come back to the game and convince their friends and family to also play the game so that the company can have more customers. This leads us to the last goal: we want to have a way for the company to make lots of money.

We assume a basic level of computer literacy in our projected demographics (four to seven year-old boys and girls). The users should be able to understand basic symbols. Another assumption that we are making is that most of these users will be playing our game from a desktop or laptop computer with readily available internet access. We also assume that the toy company wishes to maximize profits from this product.

Design Methods

Feature comparison: We created a chart with different features and pros and cons of each individual picture

Task analysis: went through every line of the task to make sure on track

Interaction logging (need to include link to site): for interview used this to check where user would click first

Personas: made to better understand our target audience

Visualization: in order to refine our product we wanted to see something in front of us

Mind mapping: this helped us get more ideas

design/making: we went through multiple iterations of making and designing the software
Requirements review: we went through the basic requirements (audience, stakeholders, goals, constraints, assumptions)

Cognitive walkthrough: we used this to figure out the different paths that the user may want to go through

Simulation: this is how we could figure out different paths and different combinations of things that can happen at the same time

Heuristic Evaluation: was kept in mind during the whole process

Application Design

To access the game, an initial purchase of the game would be required to play it. Users can also purchase additional downloadable content (DLC) in the future separately. Once users have installed the game on their desktop they are required to create an account. To create an account, users need to enter their parent's email, pick a unique username and password. While the username will be a combination of letters and numbers, the password will be a sequence of emojis. This decision was made to assist the young audience of the game more easily memorize the passwords they choose. Once a parent confirms the email for the account, users will be able to log-in and access the game.

After logging-in, users are presented with character customization screen where they are prompted to make certain decisions in regards to their in-game avatar. Users will select their avatar's name, gender, hair color, eye color, and skin color. At this time, the user is provided with default clothing for their gender respectively. Once users have completed creating their character, they will be provided a brief tutorial by a non-playable-character (NPC) that will help familiarize them with the in-game interface.

During this process the user is presented a default house that includes four rooms that are divided by two floors. These rooms will be a bedroom, bathroom, living room, and kitchen. The NPC will then show the user the in-game menu with icons that include a shopping cart, a home, a suitcase, a wrench, a group, a plus-sign with a person, a park, and a camera. These icons were chosen because they were recognizable with real-world symbols, allowing our users to assume some of what they represent.

The "shopping cart" icon allows the user to access the in-game store to purchase items for their home or clothing for their character using in-game currency earned from playing minigames. The "home" icon takes the user back to a central view of their house. The "suitcase" icon allows the user to access their inventory which presents the user with all of their owned items. The "wrench" icon will take the user into edit mode. This allows the users to add, remove, and move items throughout the house. A two-dimensional grid system will be overlayed onto the house to assist the user in spacing and moving items in each room. From the edit mode, users are also able to purchase new rooms and floors using the in-game currency. While adding a room, users can choose from a predefined template such as a game room, a garden, a pool room, or even a garage which all include preset items. The group and plus sign with a person icons involve interactions with other users regarding parties and friends. The "park" icon will allow the user to enter the public Park area when selected. Finally, the "camera" icon will take a screenshot of the current screen. This is a central area where players can interact with each other. There are also non-gameplay icons on the screen such as a "x", a "cog", and "music". The "x" icon will log

the user out of the game, the “cog” allows the user to access in-game options and the “music” icon turns on and off the music.

The user will be able to control the character’s movement by clicking on an area to move to in the grid system or by simply using the arrow keys on the bottom right-hand side of the keyboard: the character can walk left and right whether in the dollhouse or the park; when the character encounters a stairs object, can use the up/down arrow keys to move vertically on the stairs. Mini-games may also have their own specific controls that will be specified to the user (in a very clear and simplistic manner in order to convey it to the kids), which may include mouse-clicks and certain buttons on the keyboard, but will usually stick with common controls already used in the main, essential part of the game (clicks, arrow pad).

The user can interact with playable objects, NPC’s or other players by simply clicking on them. Clicking on a player will have a pop-up screen showing player info: name, relationship (friend/not friend), and awards. Similarly, clicking on an item will bring up a mini-pop up menu on the screen associated with the object. This menu will provide various details of the object and display a list of options related to the object.

Mini-games would be primarily accessed through an object in the house. If the object has a mini-game associated with it, it will give the option in the pop-up menu “Play Mini-Game”. Selecting this option will take the player to the mini-game. Other players in the dollhouse can access the object too and are welcome to join the same mini-game. A typical mini-game setting will have a lobby of players currently waiting to play the mini-game. The mini-game has an option to start, and whoever is in the lobby at the time the game is started is participating in the game. An example of this interaction would be a car object in a garage room. Clicking on the car would take the player to a lobby of a racing mini-game.

In playing the mini-games, players have the opportunity to earn rewards and earn in-game currency for the game store. Rewards can be kept as accolades that stick with the player, and these can be shown alongside the player when player details are examined.

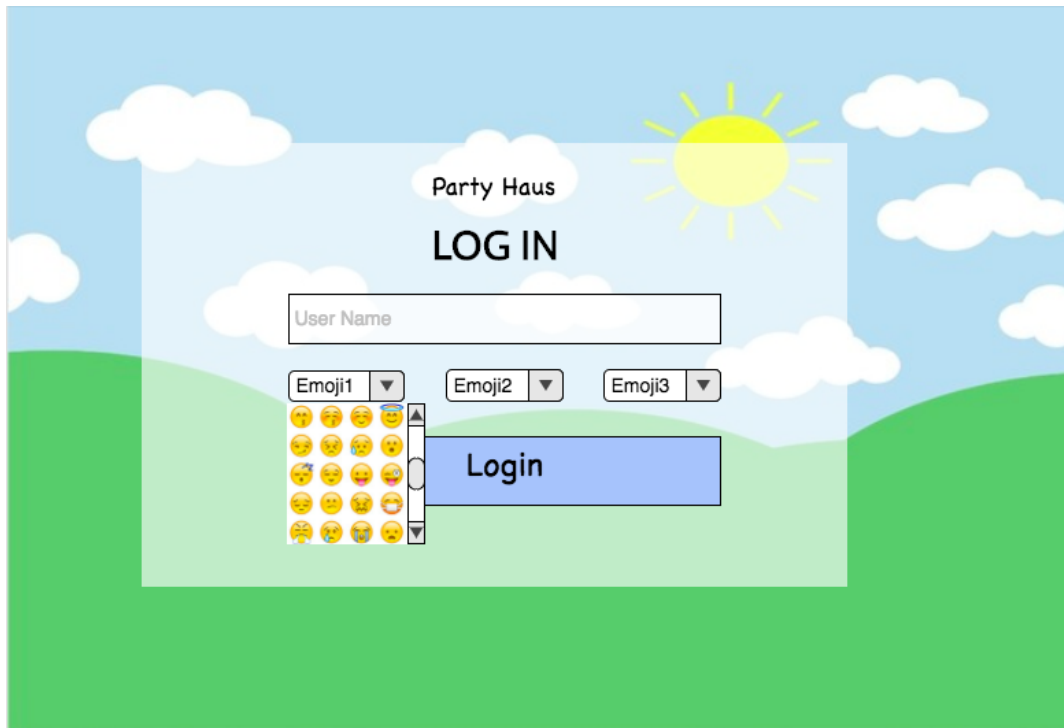
The term “party” refers to the group of players in a particular dollhouse. The players in a dollhouse consist of one party. Invitations into a party can be extended to players in the host’s friend list.

Interaction Design

The following set of mockups will briefly demonstrate the interactions between users and Party Haus, based on different user interfaces. User with an existing account will only be required to type in username at login page. While playing the game, mostly, user will be able to interact with the game by mouse-click, dragging objects and tapping on keyboard.

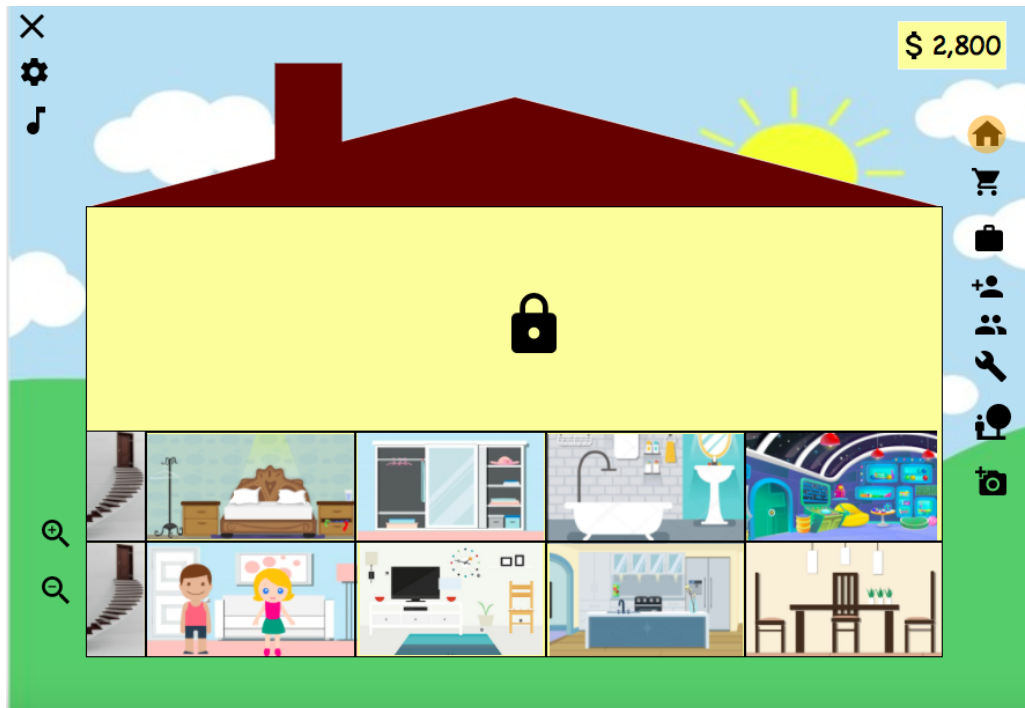
Mockups

Login



When users enter into the Login interface, they input UserName(text) and select Password(combination of 3 emojis).

House



After correct login, users enter into Dollhouse interface. User will be able to see a In-Game currency balance on top-right corner. Whenever a button is clicked by user, the button will be highlighted showing as an orange circle. Users will be able to make the characters travel inside the Dollhouse by clicking on where they want to go or tapping “arrow”s on keyboard. When users hover over an Objects with game functionality, the Object will become larger and users will be able to start a mini-game by clicking on the Object. By clicking on the characters themselves, users have the option of talking on screen(speech bubble).

On left-hand side of the interface, there are 5 buttons.

By clicking the “Cross” button, exit the game and save everything.

By clicking the “Gear” button, change user account settings.

By clicking the “Note” button, turn on/off the background music.

User are able to zoom in/out by selecting “Plus”/ “Minus”.

On right-hand side of the interface: In-Game Menu Bar

When clicking on “Home”, users will go back to the DollHouse.

When clicking on “Shopping Cart”, users will go to Store and be able to purchase items. All purchased items will be automatically saved into Inventory.

When clicking on “Bag”, users will see their in-game Inventory.

When clicking on “Plus Person”, users will add friends by name and check friend list.

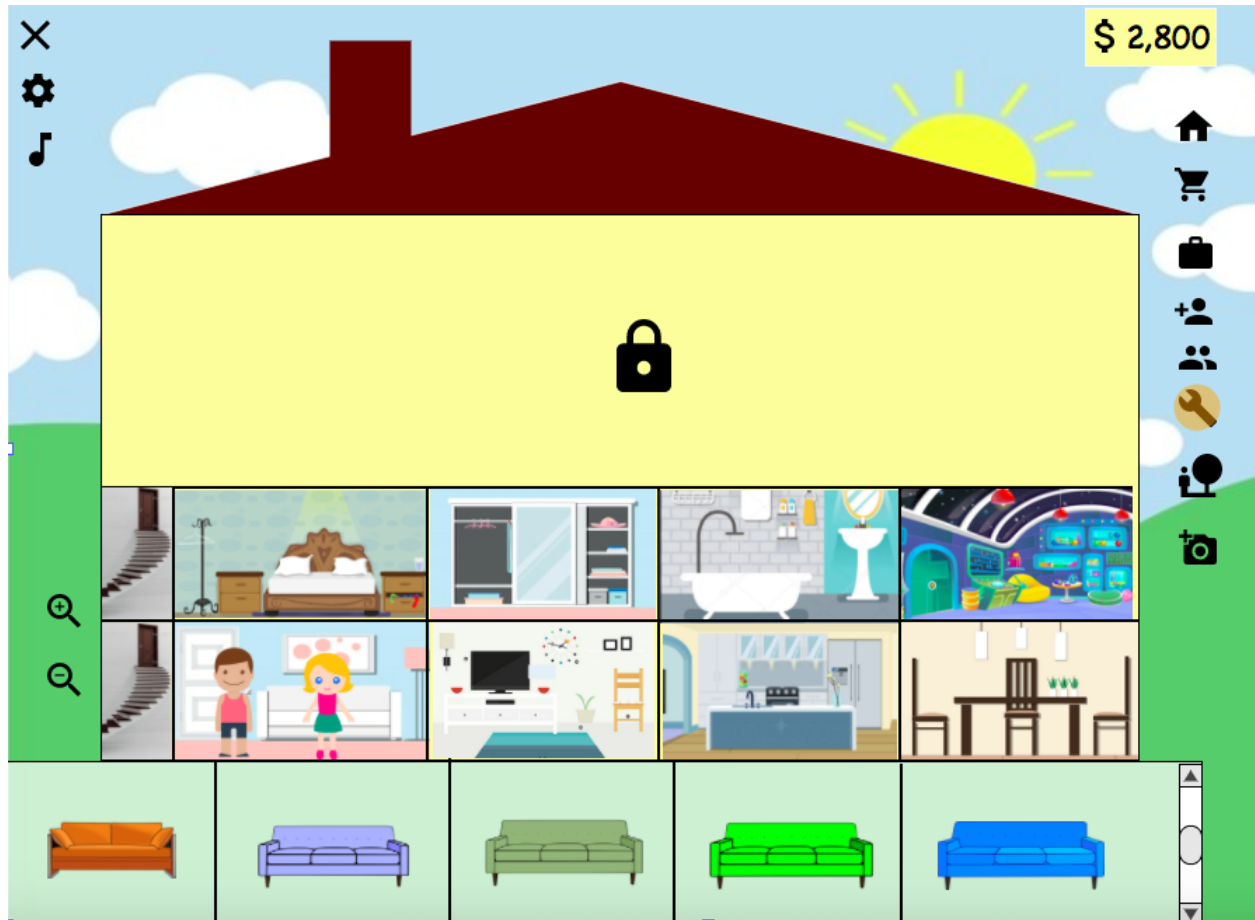
When clicking on “Multi-Person”, users will start a party by setting as public or private.

When clicking on “Wrench”, users will edit their DollHouse.

When clicking on “Person-Tree”, users will go to Park.

When clicking on “ScreenShot”, users will take a screenshot of the current user interface.

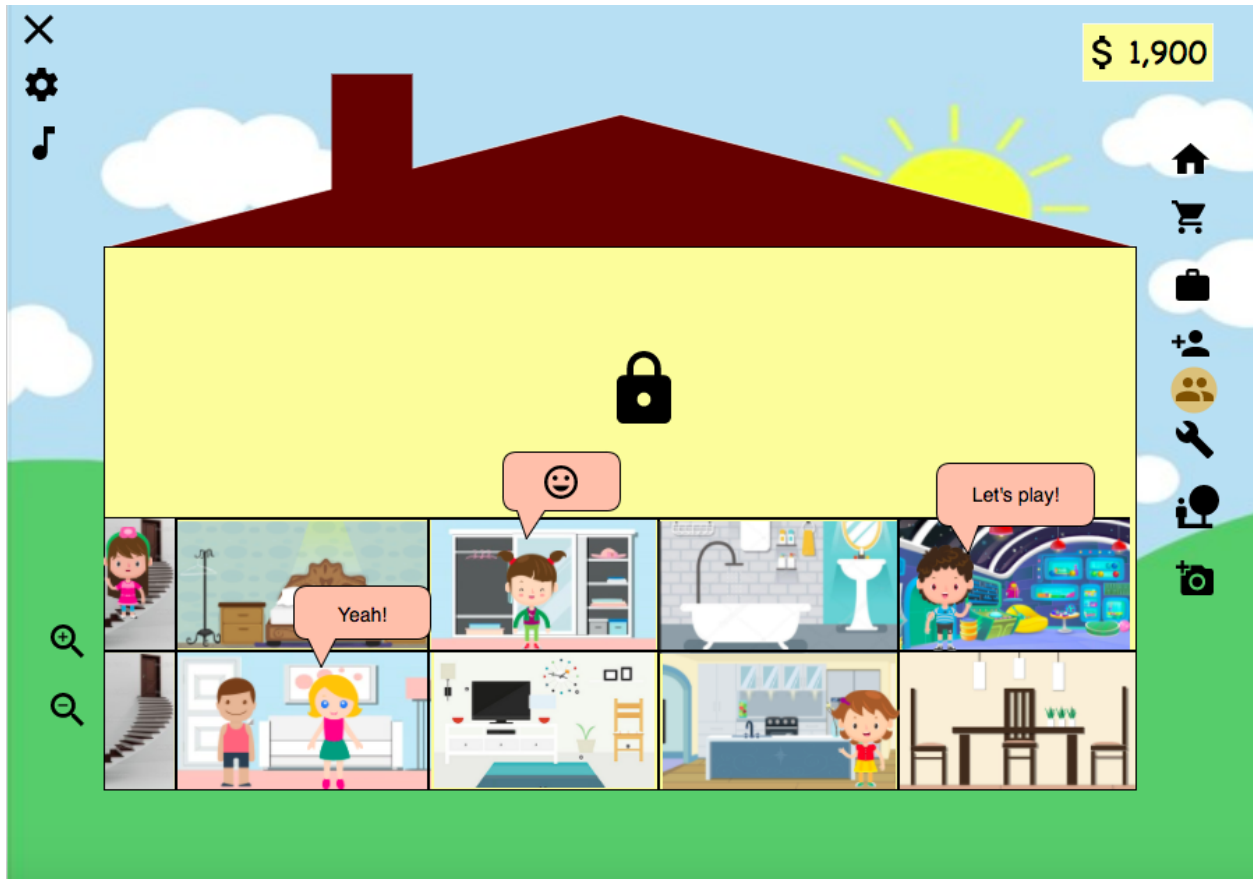
House-Editing



After clicking on “Wrench”, there will be a pop-out window on the bottom showing User’s Inventory. Users will be able to edit the objects in Dollhouse through dragging and moving actions. Users will also be allowed to interact with Inventory by adding Items from Inventory or stocking Items back to Inventory. By using the scrollbar on right-hand side, users will be able to view Inventory up and down.

After re-clicking on “Wrench”, users will exit the editing mode, and all Items will not be editable.

House-Party



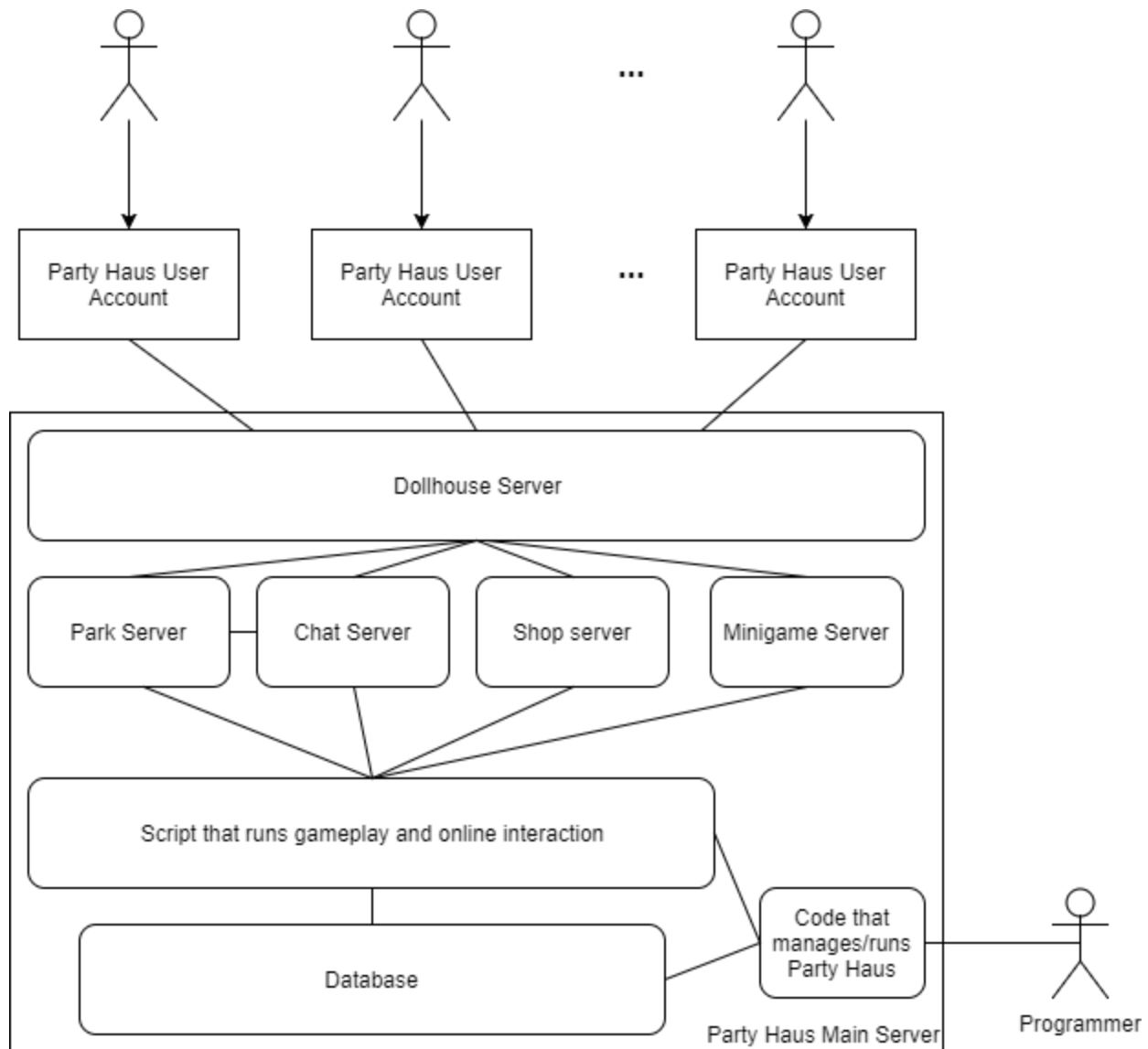
By clicking on “Multi-Person” button, user will be able to create a party. User will choose the type of the party, either public or private. While a Dollhouse is under party mode, users will be able to chat through speech bubbles and play mini-games together.

Park



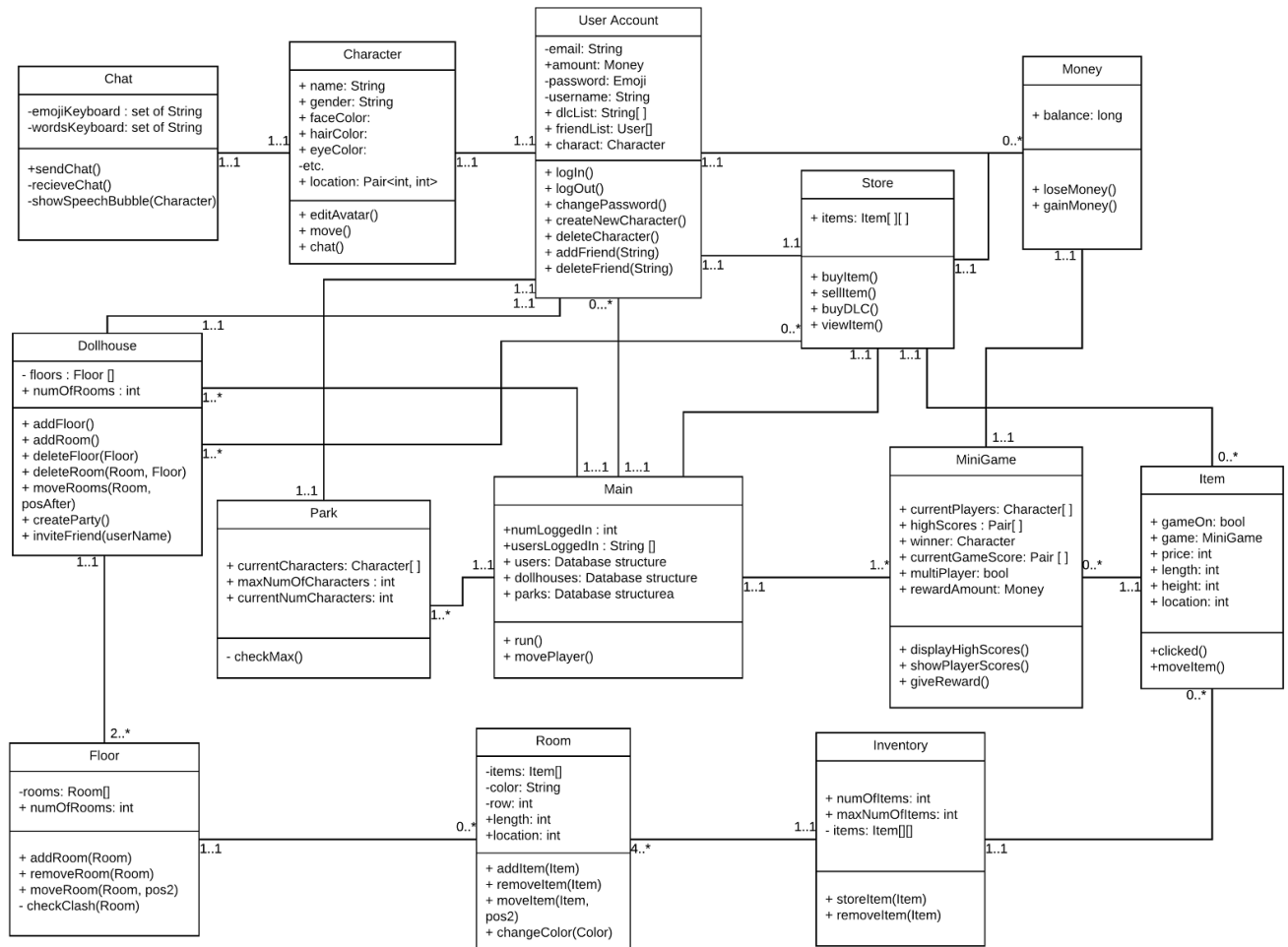
When user enter into Park, they will be able to see all the characters that are also in the Park. By clicking on a character, user will be able to see character's name and gender and have the option of sending him/her a friend request. User might receive a friend request when the "Multi-Person" button is flickering. User have the option to accept or decline a friend request. Communication with others is only allowed in the form of speech bubble, for which every user will be able to see everyone else's talking on screen. Each speech bubble will be shown for 15 seconds.

Architecture Design



The game runs on several servers which include dollhouse, chat, park, shop and minigame server. Behind all the servers is the back end code that runs the gameplay and online interaction and is all stored in a database to save each user's account/dollhouse/character information. Everything in the box is on the main Party House servers hosted in-house of the company.

Implementation Design



User Account:

Attributes:

- email: String

A String object representing the user's email

- password: Emoji[]

An array of emojis representing the user's password.

- username: String

A String object representing the user's username.

+ account: Money

Contains a Money object that keeps track of the user's in-game currency.

+ dlcList: String []

An array of String objects representing the downloadable content packs the user currently owns

+ friendList: User Account []

An array of User Account objects representing the other friends the user has in the game.

Methods:

+ login():

Signs the user into the game.

+ logout():

Signs the user out of the game.

+ changePassword():

Sends a confirmation email to the user's email for a change of password request.

+ createNewCharacter()

Makes a new Character for the user.

+ addFriend(String)

Given a username (String), adds the specified User to the friendList.

+ removeFriend(String)

Given a username (String), removes the specified user from the friendList

Character:

Attributes:

+ name: String

A String object representing the Character's name.

+ gender: String

A string object representing the Character's gender.

+ hairColor: String

A String representing the hair color of the character.

+ skinColor: String

A String object representing the Character's skin color.

+ eyeColor: String

A String object representing the Character's eye color.

+ location: Pair <int, int>

A Pair with two integers, representing the coordinates of the Character's location on the screen.

Methods:

+ editAvatar():

Initiates customization for the Character and allows the player to edit their Character's features.

+ move():

Moves the Character around the screen.

+ chat():

Sends an emoji the player selects to the screen.

Chat:

Attributes:

+ emojiKeyboard: set of String

A set of String objects representing the emojis players are available to choose from for the chat functionality. Each string is a code attached to a specific emoji.

+ wordsKeyboard: set of String

A set of String representing the words or phrases that players are available to choose from for the chat functionality.

Methods:

+ sendChat():

Sends message chosen by the player to Main.

- receiveChat():

Receives a chat message from Main.

- showSpeechBubble(Character):

Displays a specific chat message to the screen tied to the Character it came from.

Item:

Attributes:

+gameOn: bool

A boolean value indicating whether the item is obtained through a mini-game.

+game: MiniGame

Attribute specifies which MiniGame this particular Item is found (null value if not found through a MiniGame)

+price: int

An int variable indicating the price of the item.

+length: int

An int variable indicating the length of the item (necessary in relation to displaying the item's size and appearance)

+height: int

An int variable indicating the height of the item (necessary in relation to displaying the item's size and appearance)

+location: int

An int value indicating the item's location through coordinates.

Methods:

+clicked()

A method that is called when a game Item is clicked by the user. Activates the user interaction with the item, which is in a form of a mini pop-up menu screen

+movedItem()

Method is called when item is moved from place to place by the user. Allows user perform this operation.

Store

Attributes:

+items: Item[][]

Data structure that stores a list of items by their category available for purchase by the user.

Methods:

+buyItem()

Performs the transaction of the user buying an item from the store and being placed in the user's inventory.

+sellItem()

Performs the transaction of the user selling an item in his inventory to the store

+buyDLC()

Method called to perform the transaction process of the user buying in-game downloadable content with real-world currency. The process will involve verifying the purchase by sending an email to the parent (parent email stored within the user's account). From the email, the parent can confirm the purchase, be prompted for electronic payment information, and then complete the transaction by downloading the purchased feature.

+viewItem()

Allows user to view the details of the item.

Money**Attributes:**

+balance: long

A long variable indicating the amount of in-game currency owned by the user.

Methods:

+loseMoney()

Method that decreases the amount of money in the user's possession.

+gainMoney()

Method that increases the amount of money in the user's possession.

Floor:**Attributes:**

-rooms: Room[]

Array containing all of the rooms on a floor

+ numOfRooms: int

This is the number of rooms (the number can also be accessed by the size of the rooms array)

Methods:

+ addRoom(Room)

Add room to floor if there is enough free space together in order to accommodate the length of the room

Add to room its position in the floor

Else do nothing

+ removeRoom(Room)

If room in floor, remove room

+ moveRoom(Room, pos2)

If room in floor

Move to pos2 if possible to accommodate length of room

- checkClash(Room)

Check if incoming room w/position is clashing with any of the rooms already in the floor

Dollhouse:**Attributes:**

- floors : Floor []

Array containing all of floors in dollhouse

+ numOfRooms : int

Number of rooms (including rooms from all of the floors)

Methods:

+ addFloor()

Adds new blank floor to floor array

+ addRoom()

Asks for type of room and floor

Then calls addRoom on specific floor

+ deleteFloor(Floor)

Deletes the floor and removes all of the rooms in it, even if they are the rooms that the player starts with. This is only if there are more than 2 floors in dollhouse.

+ deleteRoom(Room, Floor)

Calls deleteRoom(Room) in the floor class

+ moveRooms(Room, posAfter)

if current room position is in same floor as posAfter

Call moveRoom(Room, posAfter) on that floor

Otherwise

Delete the Room in the floor its coming from, and add it to the new floor

+ createParty()

Connects to the server that this dollhouse is having a party right now

+ inviteFriend(userName)

Calls out to the server to invite the account that is connected to the userName

Room:**Attributes:**

-items: Item[]

List of items (Furniture)

-color: String

Wallpaper color

-row: int

The position on the floor to the right

+length: int

The length of the room (in roomlength)

+location: int

This is the overall coordinates of the room

Methods:

+ addItem(Item)

Add item to room if there is enough free space together in order to accommodate the length of the item

Add to item its position in the room

+ removeItem(Item)

Removes Item from items

+ moveItem(Item, pos2)

If item in room

Move to pos2 if possible to accommodate length of room

+ changeColor(Color)

Changes color to color specified

Inventory:

Attributes:

+ numOfItems: int (Number of items in a character's inventory)

+maxNumOfItems: int (Maximum number of items in inventory)

-Items: Item[][] (An array of arrays of items)

Methods:

+storeItem() (store an Item from store/dollhouse into inventory)

+removeItem() (remove an Item from inventory)

Park:

Attributes:

+currentCharacters: Character[] (An array of current characters in the park)

+maxNumOfCharacters: int (Maximum number of characters can be in the park)

+currentNumCharacters: int (Current number of characters in the park)

Methods:

-checkMax() (Check if the park reaches to the maximum number of characters)

Minigame:

Attributes:

+currentPlayers: Character[] (characters that are in the minigame)

+highScores: Pair[] (username and score are displayed together for high score)

+winner: Character (Character that won the game)

+currentGameScore: Pair [] (username and score are displayed for the minigame in progress)

- +multiPlayer: bool (boolean variable to tell whether or not the user is playing with multiple friends or by him/herself)
- +rewardAmount: Money (amount of in-game money player earns for winning)

Functions:

- + displayHighScores()
- + showPlayerScores() (shows the score of each player in the current game)
- + giveReward() (winner of the game receives in-game money)

Main: (This class is the main server that helps run the other servers)

Attributes:

- +numLoggedIn : int (Keeps track of how many total users are currently logged into the game)
- +usersLoggedIn : String [] (an array of the name of the users logged in)
- + users: Database structure (holds the data for all the information of the user's account such as what dlc's they have, their character doll customization etc.)

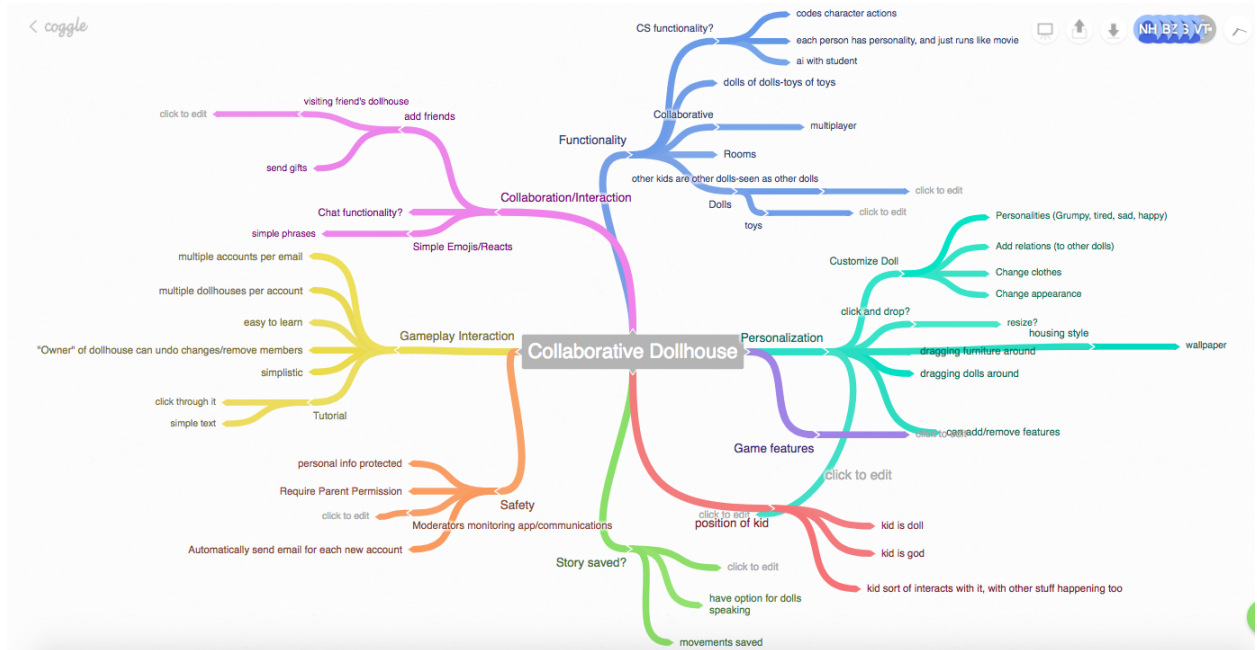
Methods:

- + run() (Method to keeps the server up and running)
- + movePlayer() (Moves player from one type of session to another. For example, when a player moves from the dollhouse to the park or dollhouse and starts a minigame)

Log

November 17, 2017

- Mind Map (Application Design)
- (img/link here) https://coggle.it/diagram/Wg8pl_kZeAABUDYT



- Participants: everyone

November 19, 2017

- Outlining key elements
- Participants: everyone

November 21, 2017

- Basic ideas, basic outline of entire system, important points (page earlier)
- Edited list from earlier
- Participants: everyone

November 28, 2017

- Preston's Algorithm:
https://docs.google.com/a/uci.edu/document/d/199PPW-XiDx4lPcIRJqI5AYmlNsn_t4Nvcs5SmPqUK48Q/edit?usp=sharing

- Priyanka's Algorithm
<https://docs.google.com/document/d/1GQNSRE3zyDjw8gK5CTeVRQ59PwF4aUA3W3L201RPgMI/edit>
- Miranda's Algorithm
https://docs.google.com/document/d/1ulvk_jjDsXhehkEdQxxR-wS8NQhryaXji_4AGnvlBM/edit
- Nelson's Algorithm:
https://docs.google.com/a/uci.edu/document/d/1wqGdKVB4bzJ_ImfLpT1sjJDhy6Q2HhJAZ3VfJmGs3wc/edit?usp=sharing
- Tim's Algorithm:
https://docs.google.com/document/d/1K1Ll7_ClvVW6oQ55uRPsovWZG58xwQG10s5A4kG-K04/edit
- Created Items to Consider list based on our ideas
 - Worked on it, not finished
- For next time: add pros/cons to list, Miranda makes mock-ups
 - Nelson ended up doing most of it
- Participants: everyone

November 29, 2017

- Made tables of decisions for the game and put it to a vote what features we want
- The idea behind it is to get everyone to agree on what the game should entail and what features we want
- Participants: everyone
- Everything done online on our own time

November 30, 2017

- Finalize decisions, features, solidify what all is going to look like, basics
- For next time: 3 personas each
- Participants: everyone

December 2, 2017

- UML Diagram, part 1
- Participants: everyone but Nelson

December 3, 2017

- Priyanka-held interview with 8 year old cousin
- UML diagram, cont.
- Reviewed mockups-edited by Miranda

- Participants: everyone

December 6, 2017

- Finish UML
 - Preston partially redesigned UML structure
- Started final document, named design
- Work on mockups, Introduction
 - Participants: everyone
- Work for next time:
 - Design Methods
 - Writers: Priyanka
 - Application Design
 - Writers: Tim and Preston
 - Interaction Design
 - Finish description, basic summary
 - Writer: Miranda
 - Architecture Design
 - Participants: Nelson
 - Implementation Design

December 7, 2017

- Continue final document, taking pieces from artifacts and adding in correct place
- Participants: everyone