The Charlotte Crisis

Game Design Document

Practice-Based Thesis

AY23/24 Semester 1

Yong Jia Yu: Designer
Choong Zhan Hong: Programmer
Shiu Wing Tung Nicole: Artist

April 12, 2024
v2024.04.12

Contents

I Game Design **4**

1 Game Overview **4**
1.1 Purpose of Document 4

2 Progression **5**
2.1 Game Structure 5
2.2 Endings 5

3 Gameplay **6**
3.1 Player character 6
3.1.1 Confidence 6
3.1.2 Stats 6
3.1.3 Experience Points 7
3.1.4 Gender Presentation/Gender Meter 7
3.2 NPCs 7
3.3 Within each stage 7
3.4 Within Dialogue 8
3.4.1 Stat Checks 8
3.4.2 Dice rolls 9
3.4.3 "Optional" choices 9

II Technical Specifications **11**

4 The Repository **11**

5 Yarn and Chatterbox Files **11**
5.1 Syntax 11
5.2 Options 12
5.2.1 Skill Check 13
5.2.2 Hidden Options 14

6 Custom Functions **14**
6.1 Player Stats 15
6.1.1 changeConfidence(int amount) 15
6.1.2 getConfidence() 15
6.1.3 changeGenderMeter(int amount) 15
6.1.4 getGenderMeter() 15
6.1.5 addExperience(int amount) 15
6.1.6 incrementStage() 15
6.2 NPC Relationships 16

6.2.1 changeRelationship(string npc_name, int amount) 16
6.2.2 getRelationship(string npc_name) 16
6.2.3 setNPCKnows(string npc_name) 16
6.2.4 getNPCKnows(string npc_name) 16
6.3 Skill Checks 16
6.3.1 diceRoll(int stat) 16
6.3.2 calculateOdds(int stat, int threshold) 17
6.3.3 skillCheck(int stat, int threshold) 17
6.3.4 visitedNode() 17

Part I

Game Design

1 Game Overview

The Charlotte Crisis is a single-player narrative role-playing game. The player controls Charlotte, a male-to-female crossdresser who must convince her family to accept her feminine identity. Charlotte interacts with various NPCs through dialogue. Charlotte is able to make branching dialogue choices, with certain options requiring her stats (Intellect, Charisma or Empathy) in skill checks. Charlotte can also determine her presentation (male/female), with the resulting dysphoria affecting her stats. The game ends when Charlotte overcomes three confrontations with family, which serve as boss fights. This game is appealing for RPG fans and players who enjoy interactive storytelling.

1.1 Purpose of Document

This document contains several sections. This first part acts as a Game Design Document, and contains information on the game's overall design and mechanics.

The subsequent parts serve as a reference for the current developers. The technical section (Part II) contains information on the game engine(s) being used, guidelines to follow, code standards, and reference for the developers.

The Charlotte Crisis is created as part of a Practice-based Honours Thesis. The research and reflection for the thesis project will not be included in this document.

2 Progression

We aim to make a story-based game about gender presentation, where players make meaningful choices on how they present themselves and interact with NPCs.

2.1 Game Structure

The game comprises 3 **levels**. Each level will be divided into 4 **stages**:

- Class
- CCA (Co-curricular Activity)
- Hangout: choose one NPC from Class/CCA to talk to, at length
- Boss fight: unique NPCs that pose a greater challenge

In total, players will access up to **12 stages**. Player experience may vary as some scenes, for example, the hangouts, are only accessible if the player makes certain choices (that affect relationships with NPCs).

2.2 Endings

Depending on player choices affecting their relationship with the two potential final bosses, 5 possible endings can be attained.

- Final boss Charlotte: spare Charlotte
- Final boss Charlotte: kill Charlotte
- Final boss Mother: Understanding
- Final boss Mother: Broken ties
- Final boss Mother: Mother wins

3 Gameplay

3.1 Player character

The player character has two identities: Charlotte (Female-presenting) and Charles (male-presenting), which the player may choose from at the start of each stage.

The player can move freely around their environment (left and right), and interact with NPCs through dialogue.

3.1.1 Confidence

The player is given **Confidence**, a percentage bar that diminishes and replenishes with negative and positive interactions respectively. If **Confidence** reaches 0 percent, the player incurs massive debuffs to Stats (see below). Confidence ranges from 0 to 100. The player begins with 50 confidence.

3.1.2 Stats

The player also has Primary Stats: **Intelligence (INT), Charisma (CHA), Empathy (EMP)** that influence the consequences of dialogue options. For example, choosing an **Empathy** dialogue choice, while not having enough **EMP** to pass its minimum threshold, may cause consequences like loss of Confidence, or lowered NPC Relationship.

Players can set their Stats at the start of every stage, through Stat points earned in dialogue and by completing levels, as well as through different outfits.

Stats are utilised in dialogue (See Stat Checks).

- Confidence: equivalent to hit points. Players start with a maximum of 100. Can be depleted or replenished through interactions.
- Intelligence (INT): required for logical, factual dialogue options like:
  - "Your argument isn't logical, it's emotional."
  - "Gender is a social construct"
  - "Everyone has the right to be their most authentic self"
- Charisma (CHA): required for dramatic, suave dialogue options like:
  - "Crack a joke!"
  - "Don't worry, we've got a plan."
  - "I'm not letting others' opinion affect me."
- Empathy (EMP): required for emotional, "kind" dialogue options like:
  - "Tell me what you're going through."

- "Please, just see it from my point of view."
- "I can relate to you."

3.1.3 Experience Points

The player can gain experience points **(XP)** from successful NPC interactions. They may use XP to restore Confidence, or increase their Stats, at the end of every stage. 50 XP will be required for levelling up once, upon which one Stat point will be given.

3.1.4 Gender Presentation/Gender Meter

The gender meter also serves as a makeshift "Relationship meter" between the player and Charlotte, where a feminine value on the gender meter reflects a better Relationship, and a low enough value will trigger a boss fight with Charlotte. The gender meter ranges from 0 to 100. The player begins with it set to 50.

Players' choice of gender presentation at the start of each stage, and their responses to NPCs' dialogue (such as picking gender affirming dialogue or not) will affect their gender presentation (and internalised identity).

3.2 NPCs

Non-player characters may be approached for conversation. They will each have their own stats, reflecting their proclivity towards certain talking points and conversation styles. For instance, to succeed in interactions with an NPC bearing a high INT Stat, players can either increase their own INT Stat to exceed the NPC's, or leverage another of the NPC's lower stats (EMP/CHA).

NPCs will also bear a **Relationship meter** towards the player, which changes according to how closely you agree with their views. For example, a statement that offends an NPC's personal values will decrease their relationship with the player character. The Relationship meter will affect who you can invite for the Hangout stages (you'll need at least neutral Relationship with them).

The relationship values are as follows:

- Good relationship: 70 - 100
- Neutral relationship: 31 - 69
- Bad relationship: 0 - 30

3.3 Within each stage

Players start the stage in Charlotte's bedroom. They will decide if they want to present as female (Charlotte) or male (Charles). A random dice roll made after this choice (symbolising dysphoria) will award bonuses/debuffs to player statistics.

Players will then enter the daily interaction (Class/CCA/Hangout), exploring the game environment to interact with NPCs. By engaging in conversations with them, they will progress through the game's story. Conversations can either be with a single NPC or with multiple NPCs, with each stage ideally having one of each type.

3.4 Within Dialogue

The game and its narrative is primarily driven by dialogue. Players will be presented with dialogue options that affect their conversation partner's resultant dialogue. Certain dialogue options will affect relationships with other characters, and may affect the player's confidence as well.

Some choices will be attached to a Stat value. The player will be informed on the stats required for that dialogue option, and if the player's corresponding Stat value falls below the required value, the dialogue choice will "succeed", else, it will "fail" (See Stat Checks).

Dialogue choices can affect Confidence, Relationship, gender and can give XP.

3.4.1 Stat Checks

Some dialogue options come with a requisite amount of stats (**INT**, **EMP**, or **CHA**). An example dialogue is as follows:

> -> YOU: "But what is a guy?" (INT: 3)
>
> ANTHONY: "What?"

In this case, the stat requirement is 3 or more **INT**. If the player has enough **INT**, the following dialogue will play out:

> YOU: "I'm a crossdresser, so I go between guy and girl. It's that easy."
>
> VERA: "Love that for you! Charles and Charlotte."
>
> ANTHONY: "Does anyone not see how weird this is?"
>
> *He is met with silence.*
>
> ANTHONY: "Guess I'm the only one with any common sense left."

Otherwise, the following will play out for this snippet. Note that if the player chooses to proceed with options despite not having enough stats, they may not benefit or even be harmed by the interaction (such as by loss of confidence).

> YOU: "Yeah. Why can't a guy wear dresses and makeup too?"
>
> ANTHONY: "I'm not gonna sugarcoat it. That's really weird."
>
> ANTHONY: "How many guys do you see do that? I certainly don't."
>
> VERA: "Hey, let Charlotte wear whatever she wants."
>
> ANTHONY: "Why are you referring to him by that name? His name is Charles. Let's not feed into delusions here." (-5% confidence)

Stat checks can be automatically done by the game, showing dialogue options that would otherwise remain hidden if the player does not have enough stats.

3.4.2 Dice rolls

Some dialogue options will have different outcomes depending on a "dice roll". This mechanic adds uncertainty to gameplay, just as real-life conversations are unpredictable based on the conversation partner.

The mechanic adds the current player Stat required for the option with the outcome of one 6-sided dice rolled once. This result is compared against the required threshold for the Stat. If the outcome of the dice roll is higher than the threshold, a positive or desired outcome will commence in dialogue. If it is lower, a more undesirable outcome will be shown instead.

An example is shown below:

> -> "You've got some hot guys on that sheet of paper too." (CHA check)
>
> Nadia: "Huh? What? It's nothing! I'm just experimenting!"
>
> You: "Come on, I won't judge."
>
> *(CHA >= 6)*
>
> Nadia: "Okay, fine... I like anime. Especially those with hot guys..."
>
> You: "You should take pride in that! It's your own style."
>
> You: "I think the drawings look good! Both your husbandos and yourself."
>
> Nadia: "Thank you... it's rare to hear that. Usually people just judge me." (+5% Nadia relationship)
>
> *(CHA < 6)*
>
> Nadia: "Ugh... I'm not comfortable sharing that."
>
> Nadia: "Please... don't intrude on my work. I'll show it when I'm ready."

3.4.3 "Optional" choices

To allow players flexibility on how much they want to read before making a decision, some choices will be made optional. These choices, if selected, will return the player to the original question that prompted them, instead of continuing the narrative. An example would be:

> Nadia: "Oh, I'm just drawing myself."
>
> Nadia: "I'm still in the work-in-progress stage, though, so bear with me!"
>
> -> "Hmm. Why did you draw in anime style?"
>
> Nadia: "Well... I guess it's easier, right?"
>
> Nadia: "After all, it was designed for 2D artists."
>
> You: "That's fair."
>
> **-> "It's okay, I guess. Not much else to say."**
>
> Nadia: "Alright then."
>
> Nadia: "Have fun with your own artwork too."
>
> **-> "You've got some hot guys on that sheet of paper too."**

Nadia: "Huh? What? It’s nothing! I’m just experimenting!"

You: "Come on, I won’t judge."

The first choice will be made optional: players can select it to gain information about an NPC through exposition. However, players can also choose either of the options below to pass judgement on the NPC, and advance the story.

Part II

Technical Specifications

This part documents the tools used, and the guidelines to use them consistently within the project. The links lead to documentation for the respective tools. This part will be useful for programmers especially, but also the team to understand how the game works across the tools being used, and how to standardise the assets of the game.

For the prototype, The Charlotte Crisis is built on GameMaker. Branching dialogue is coded with Yarn Spinner. We also used Crochet editor, which is not necessary but simplifies the process of writing branching dialogue in a visual, node-based editor. Our Yarn files were then integrated into GameMaker with Chatterbox.

4 The Repository

The Charlotte Crisis is maintained on a GitHub repository: charlotte-crisis/charlotte.

The GameMaker project is stored in the directory `Charlotte-Crisis`. Within this directory, the `.yarn` files containing the story's narration and dialogue are stored in `datafiles` for GameMaker to access.

5 Yarn and Chatterbox Files

5.1 Syntax

The Yarn files allow branching dialogue to be written in plain text. Regular dialogue is in the format of `SPEAKER: Speech goes here`. To note, any text that comes before the first colon `:` in Yarn is demarcated as the speaker's name. Subsequent colons in the same line can be used normally in dialogue.

The `SPEAKER:` is to be denoted in CAPITAL LETTERS. When the player is speaking, use `YOU:`. The narrator, or otherwise non-spoken dialogue like the player's inner monologue, is indicated with no speaker. Spoken dialogue is represented with double inverted commas `"like such"`.

An snippet of Yarn script following these conventions is as below.

```
VERA: "That logo... is that what I think it is?"
YOU: "...what do you mean?"
VERA: "It's Very Gender."
Why did she just emphasise those two words?
VERA: "So. Tell me about it. What's the logo mean?"
YOU: "It's nothing."
Vera raises an eyebrow.
```

In the game, narration as well as the player's inner monologue are to be italicised. The above dialogue it will appear as such (each line is displayed separately).

VERA: "That logo... is that what I think it is?"
YOU: "...what do you mean?"
VERA: "It's Very Gender."
Why did she just emphasise those two words?
VERA: "So. Tell me about it. What's the logo mean?"
YOU: "It's nothing."
Vera raises an eyebrow.

5.2 Options

Dialogue can include options through the arrow notation, `->`. Sample dialogue with three options looks like this in Yarn:

```
Joanne: "As a guy, what kind of gift would
you appreciate most from your girlfriend?"
    -> Headphones
    -> Limited edition album
    -> A pair of sneakers
```

The player will see the text before the three options, while being able to select from the three. Note that due to how Chatterbox works, yarn files **must have a visible line of speech before options**, or else the options will not display. This means that options (indicated by arrow notation) cannot come after code like `<<addExperience(10)>>` or be the beginning lines of a new node.

Options can branch into text that only appears when you select the option.

```
You: "How was your weekend?"
Joanne: "Good! Just watched a movie with my boyfriend."
-> "Ooh! What movie?"
    Joanne: "Parasite. It's pretty good."
-> "At the cinema?"
    Joanne: "No, at home. We don't really like going to the cinema."
    Joanne: "It's a bit expensive."
```

If you select the first option ("What movie?"), you will only see the line that follows ("Parasite..."). Branches follow indentation. After an option, whichever follows must be indented. You can also have nested options.

At some point, some dialogue may check against your stats. There are two types, **Skill Check** and **Hidden Options**.

5.2.1 Skill Check

This is where the option can be chosen but requires a dice roll along with a certain amount of stats, which results in varied dialogue.

The outcome is calculated by the outcome of a 6-sided die added to the skill to check.

Unless otherwise stated, a success in a skill check against a threshold will be if the outcome is **greater than or equal to** the threshold.

To format the skill check, we make use of the **Metadata** feature of Yarn and Chatterbox. Metadata is indicated with a hash symbol `#`. We use metadata to be able to detect skillchecks.

For skill checks, please suffix them in the Yarn file with three pieces of metadata: `#skillcheck` to indicate it is a skillcheck, followed by the **skill** and the **threshold**.

For example, a 3 INT check will look like this:

```
YOU: "Well, ants aren't all bad, right?" #skillcheck #int # 3
```

Note that for numerals, **a space is required between the number and the hash symbol**. This is a limitation of Yarn or Chatterbox. The metadata will show up as blank otherwise.

Within GameMaker, there is a function to automatically prefix the option with `[<STAT> <THRESHOLD>]`, which indicates to the player that this is a skill check. The option is also suffixed with the percentage chance of success (see section 6).

The above code will be rendered, for example, as such in the game:

> YOU: [INT 3] "Well, ants aren't all bad, right?" (66% success)

The outcome differs if the player rolled 3 or more on the skill check. A successful check leads to the following dialogue:

> ANTHONY: "What?"
>
> YOU: "Yeah. Hardworking, can lift 100 times their body weight, all that stuff."
>
> ANTHONY: "Okay, and?"
>
> YOU: "That's just another way to interpret your art."
>
> YOU: "Why see only the bad side when there's a good side to it too?"
>
> ANTHONY: "I guess so..."
>
> *(+5% Anthony relationship)*
>
> ELEANOR: "Exactly. Okay, what about you, what did you draw?"

With a roll of 2 or less, the below dialogue plays out instead. Note that the successful check led to an increase in the relationship with the NPC.

> ANTHONY: "What?"
>
> YOU: "Uh... ants are..."
>
> ANTHONY: "Trash. Pests. Like me."
>
> *You feel like you haven't contributed to this conversation. Moving on...*
>
> ELEANOR: "Okay, what about you, what did you draw?"

5.2.2 Hidden Options

These are options that are only revealed if the player has enough stats. If the player does not have enough stats, the option simply will not be shown and cannot be picked.

An example list of options can be seen below. The last option, prefixed with `[CHA]`, means that the player had sufficient *Charisma*, and can pick the option. Otherwise, it would not have been shown.

> *You sense there is an awkward silence in the air.*
>
> -> Change the topic.
>
> -> Say nothing.
>
> -> [CHA] Crack a joke to relieve the tension.

6 Custom Functions

Functions can be added into Yarn via Chatterbox to change certain things in GameMaker. This section documents the functions that are used within Yarn that affect things in GameMaker.

Custom Functions are added from GameMaker code using Chatterbox Function `ChatterboxAddFunction`.

An example line in GameMaker:

```
ChatterboxAddFunction("setGender", set_player_gender);
```

As described in the documentation, the first parameter supplied is the function name in Chatterbox (`setGender`), and the second is the function as defined in GameMaker (`set_player_gender`).

The function can then be used in Chatterbox as such:

```
<<setGender(0)>>
```

In this example, the player's choice in Chatterbox will change their gender.

A list of the functions added to Chatterbox is found below with respective description and usage guide.

6.1 Player Stats

6.1.1 changeConfidence(int amount)

Description: This changes the player's confidence level. Negative numbers are accepted. The player's confidence level will never go above 100 and below 0, and this will be handled by the function.

Usage: `<<changeConfidence(-5)>>` will reduce the confidence by 5%.

6.1.2 getConfidence()

Description: This will return the confidence as an integer (0 to 100) which are typically used for checks, such as boss outcomes.

Usage: `<<getConfidence()>>` will return the number 50 if the player's current confidence is 50.

6.1.3 changeGenderMeter(int amount)

Description: This is quite similar to the above, instead dealing with the player's gender meter.

Usage: `<<changeGenderMeter(-5)>>` will reduce the Gender Meter by 5%.

6.1.4 getGenderMeter()

Description: This will return the gender meter as an integer (0 to 100) which can be used for checks.

Usage: `<<getGenderMeter()>>` will return the number 50 if the player's current gender meter is 50.

6.1.5 addExperience(int amount)

Description: This will increase the player's experience (XP). If it goes above the level up threshold, the player will level up.

Usage: `<<addExperience(5)>>` will increment XP by 5.

6.1.6 incrementStage()

Description: This is to be used at the end of Class, CCA, Hangout (or no hangout), and Bosses, to increment an internal variable that tracks player progress. Goes from 0 to 11.

Usage: If the player is in CCA1 (stage 1), `<<incrementStage()>>` will increment the counter to 2 (hangout).

6.2 NPC Relationships

6.2.1 changeRelationship(string npc_name, int amount)

Description: This will change the relationship meter of the character as denoted by the `npc_name` parameter. As relationship is from 0 to 100%, this function handles cases where relationship goes below 0 or above 100.

Note that NPC names are initialised. Chenjie is written as `"CJ"`.

Usage: `<<changeRelationship("CJ", -5)>>` will reduce the relationship with `Chenjie` by 5%. Note the negative sign.

6.2.2 getRelationship(string npc_name)

Description: This will return the relationship value of the stated NPC (0 to 100).

Usage: `<<getRelationship("CJ")>>` will return Chenjie's relationship level. If the relationship level is 25%, it will return 25.

6.2.3 setNPCKnows(string npc_name)

Description: This will update the variable tracking if the NPC denoted by `npc_name` knows about *Charlotte*.

Usage: `<<setNPCKnows("CJ")>>` will update that Chenjie knows about Charlotte.

6.2.4 getNPCKnows(string npc_name)

Description: This will return true if the NPC indicated knows about Charlotte, and false otherwise. **Usage:** `<<getNPCKnows("CJ")>>` will return whether Chenjie knows about Charlotte. Example: To show a line of dialogue if the NPC knows, do this:

```
Chenjie: "I know your secret" <<if getNPCKnows("CJ")>>
```

6.3 Skill Checks

The below skill check functions will automatically include any debuffs invoked by having a low confidence (below 30). There is thus no need to make additional checks, and these can be used as-is regardless of situation.

6.3.1 diceRoll(int stat)

Description: This will generate a random number (1-6) and add it to the stat. In this case, please use the stats as stored in the Yarn file, which are invoked by using a prefix of $, such as `$cha` for the charisma stat.

Usage: `<<diceRoll($cha)>>` will return the value of a dice roll added to the current charisma stat. This is to be used in conjunction with skill checks.

6.3.2 calculateOdds(int stat, int threshold)

Description: This will return the percentage chance as a number (0 to 100) to succeed a skill check with the stat's current amount and the threshold to beat. This is used similarly to the diceRoll function above.

Usage: `<<calculateOdds($cha, 5)>>` will return the chance of succeeding a charisma skill check of threshold 5. E.g. if your CHA stat is 1, then this will return 50 (rolls of 4, 5, or 6, i.e. 50% of possible outcomes, will result in a success). Note that it returns a number, without the percent sign.

6.3.3 skillCheck(int stat, int threshold)

Description: This will perform a skill check for you and return true if it's a success. This will, in essence, perform `diceRoll(stat)` and compare it against `threshold`. This is not used as much, as you may need the diceroll outcome value

Usage: `<<calculateOdds($cha, 5)>>` will return the chance of succeeding a charisma skill check of threshold 5. If you

6.3.4 visitedNode()

Description: This is a more powerful version of the visited function, which uses this Chatterbox function.

Usage: In any Yarn line, `<<visitedNode("1", "class1.yarn")>>` will return the number of times the node "1" in "class1.yarn" has been visited (does not have to be the current Yarn file).