Detective Simulator

Motivation:

Detective simulating games and debugging in programming are quite similar as they both require one's ability to analyze details and spot errors. This game, as all serious games do, plans to educate the players, specifically, we aim to train the players' critical thinking ability.

Objective:

This is a simulation/game based on real-life murder cases that relies heavily on the player's deductive reasoning to win. The player plays the role of a chief of police that intends to find the culprit behind a series of murder sprees.

At the beginning of the game, an inhabitant is randomly selected as the murderer and starts murdering inhabitants. As all inhabitants differ in some way, (in attributes that represent them in the relation schema occupation, gender etc.) the murderer would take a different mode of action in this simulation. (i.e. if the murderer happens to be a student, he/she will then be limited to kill after school or face the risk of appearing as someone behaving anomalously were the player chose to interview the murder's school)

In each round of the game, the player is given police records on all inhabitants of the area (the entire database) as well as murder cases and incidents encountered by the police officers. Through analyzing the deaths of the inhabitants (cause of death, time of death, location of corpse, and etc.), testimonies from eyewitnesses, and relationship web of victims (potentially done using recursive query) the player is asked to take actions to catch the murderer. The player should make use of all the information gathered to deploy police resources such as patrols and checkpoints to intercept the culprit while minimizing the inconvenience of law-abiding citizens.

The simulation ends when the player successfully arrests the culprit or when the player was unable to locate the culprit in a certain number of rounds.

Validity:

In reality, investigators work with a relationship web of suspects and have access to a database containing information on registered residence. It is natural to recreate, as accurate as possible, the tools that are available to an investigator using a database schema. ER diagrams are also very useful in representing relationship web between inhabitants. Also, as our simulator will be working with a large number of inhabitants, there will be a large quantity of data to be managed. For these reasons, we believe that an Entity-Relational model using SQL is needed to effectively manage the data that we will be working with.

This project will require a database potentially as complex as UNIVERSITY and COMPANY. To list a few, we will need to represent many relations in this game. To list a few that come to mind at the moment, we will need a table for: Inhabitant characteristic, Victims, details of death, geographic location details, inhabitant relationships with victims, occupation of inhabitants. As we need to represent this information to the player, the queries that need to be made will not be simple as well. For one, we will need to represent a relationship web of all inhabitants in town, which would potentially require a recursive query on tables containing relevant information. We have not figured out all the technical details of this project, which means the complexity of this project only increases from here.

Although SQL databases are usually not used for the real-time part of video games, it still has many use cases for storing persistent game data such as game rule data and game save data. While these two uses don’t make the most of database techniques as they are mostly just using databases to dump the game data to disk, some games do use relational databases to manage

complex relations. These examples include tech trees in Civilization 6 and football players in football manager. Therefore, we believe SQL databases are suitable for our game as it would be a turn-based game that doesn't require real-time simulation and that the operations involved in the game mechanics are highly relational.