AN SURVIVAL KNOWLEDGE TRAINING PLATFORM TO SELF-RESCUE FROM CAMPUS SHOOTING USING 3D MODELING AND MACHINE LEARNING

Congyu Zhao[1], Suraj Singh[2]

[1]Northwood High School, 4515 Portola Pkwy, Irvine, CA 92620
[2]Computer Science Department, California State Polytechnic University, Pomona, CA 91768

ABSTRACT

School accidents have still been a major problem that happens frequently over decades [1]. Safety hazards are also the problem that parents and teachers were worried about, especially campus fire [2]. Schools are the main places where we study and live, and are also special places where a large number of minors gather. How to do a good job in campus fire safety is not only related to the normal order of education and teaching on campus, but also to the safety of teachers and students. the vital interests of households and the future of the motherland. According to statistics, more than 80% of school fires are caused by human factors, and classrooms, student dormitories, restaurants and other living places are more prone to fires. Habitual violations of regulations have become one of the main factors that cause fires [3]. However, base on all those type of question, this paper designs an game for preventing campus fire and other types of safety hazards might appearances at school. Our design builds upon on unity engine and C# script [4].

KEYWORDS

Settings, Menu, Opinions, FPS, TPS

1. INTRODUCTION

My topic is about avoiding dangers caused by natural disasters [13]. It can be used in multiple school simulations and popularly engage and warn students and teachers by reducing the difficulty certain types of accidents could have due to carelessness. It could benefit school and let people notice what to do when this type of accident has happened. It will be helpful by knowing and telling what to do and increase the chance of survival in different ways.

Some of therelated tool I use the most is definitely unity. And it provides a platform that my game builds upon. The second technique and system that has been proposed is github [5]. This app is used to store everything for you. You can easily find out what is going on in your code and update the program while you update your game which is really helpful. You can also play a preview version of the game too! It also provides many important information and knowledge on this website. It has a strong database of search and sharing your program that might benefit other users. There are also other tools which are rarely the case in practice. Their implementations are also limited in scale, with samples given for the importing assets store and spectacle technique that form unity itself. The assets store really benefits the creator in many many ways. You don't have to spend your time on creating new material and learn to draw or something just for creating models like chairs or desks. You could find cheap or even free assets like that and importing It to

David C. Wyld et al. (Eds): SIPM, ITCA, CMIT, FCST, CoNeCo, SAIM, ICITE, ACSIT, SNLP -2023
pp. 131-138, 2023. CS & IT - CSCP 2023 DOI: 10.5121/csit.2023.130914

the game by downloading. Which saves your time wisely and you could make your own copy and benefit others as well by sharing your work. Lastly is the specified and very smart technology that unity has. Unity has very strong compatibility on platforms such as Apple, Windows,pc,iPhone [6]. Second of all, It have timeline or level generator for you to use and advance your game. It creates better upper or lower limits for developing and advancing your game due to those things. Third, unity have variety coding language for you not feeling too stressful that you were able to select as your favor, It further increase the stability and simplicity for have facilitating inviting more volunteer to involve that made the unity became most popular and most well known mainstream game engine compare to others.

In this paper, we follow the same line of research. Our goal is to limit the damage caused to people from School accidents. Our method is inspired by a developed game system and one of the most popular methods in students nowadays. There are so many good features that make this maneuver interesting.

The rest of the paper is organized as follows: Section 2 gives the details on the challenges that we met during the experiment and designing the sample; Section 3 focuses on the details of our solutions corresponding to the challenges that we mentioned in Section 2; Section 4 presents the relevant details about the experiment we did, following by presenting the related work in Section 5. Finally, Section 6 gives the conclusion remarks, as well as pointing out the future work of this project.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Limited Skill

There are so many great ideas but we only have limited skill to range, tries to make it real but still difficult to implement, always something to improve on and always with some unsatisfying. But I think that is the point of making a great game, always brainstorming, Game design is like making a movie. A great game can see a clear design path, the system from beginning to end, and players can have the pleasure of seeking growth, which is a perfect fusion, just like the classic sequence of movies, story, system.

2.2. The difficult Level of Game Development

There was a time when I even had the illusion of "difficult to implement = fun", and when I saw a reference game, I felt like "it's very difficult to make this function of this game (for me) / the workload is terrible, So it's a good game" idea. After newbies get into game development, they feel that this is amazing and that is amazing, but it may be just a fuss for experienced and well-informed practitioners.

3. SOLUTION

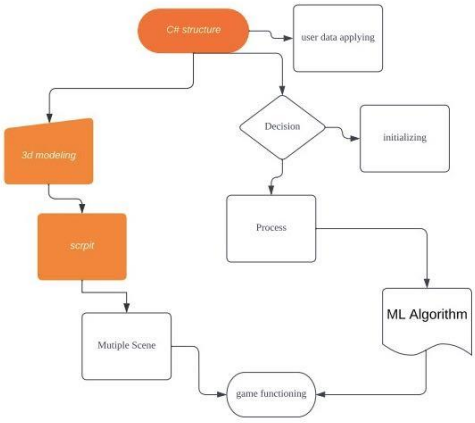

Figure 1. Overview of the solution

I implemented each component one by one in order to accomplish the fundamental setting of the project, by starting to build the random level building and npc which is the character. That the npc were able to have a conversation with the player. And you have multiple answers to select and npc provides you different feedback.

We use C# script or unity coding that C# programs are used for implementing various concepts and the coding is written in visual studio code [7]. Architecturally, Visual Studio Code combines the best of web, native, and language-specific technologies. Using Electron, VS Code combines web technologies such as JavaScript and Node. is with the speed and flexibility of native apps like unity I used to build my game.

Figure 2. Screenshot of the game

This is an NPC conversation [8]. The algorithm I used is that the simple serializefield is connected and referring back to unity that you have a space where to build up settings, similar to creating your own component. And the public means the data you are available to change in game data, and it shows in game. And the private is like the initial setting that you weren't able to change, and its database located at.

```
[SerializeField] NPCConversation conversation;
[SerializeField] GameObject player;
private Vector3 orginalDestination;
public UnityEvent OnDialogeStart = new();
public UnityEvent OnDialogeEnd = new();
private bool isCurrentDialogeActive = false;
[field: SerializeField]
public bool IsNPCActive { get; private set; } = true;
```

Figure 3. NPC coversation

And the next code is based on the task system the game has. The task system means that you have to accomplish every task in order to go to the next level/plot. There are also some tasks triggered by the conversation over npc characters. In this part of the code, we used boolean to run it. Boolean operator is "OR," which broadens your search [9]. This operator is used to express that as long as one of two or more conditions are met, the value of a specified query is true or false. The first part is to ensure if you completed the previous tasks that it is able to add new tasks eventually, and the second part is to remove the task after you finished the tasks and show mark off in the task board. And the last thing is to check the task if it is completed.

```
public bool AddNewTask(ITask NewTask)
{
    if (currentTask.Contains(NewTask)) return false;
    currentTask.Add(NewTask);
    onTaskAdded?.Invoke(currentTask.Count + completedTasksCount, NewTask)
    return true;
}

public bool FinishTask(ITask Task)
{
    if (currentTask.Remove(Task))
    {
        completedTasksCount++;
        onTaskCompleted?.Invoke(currentTask.Count + completedTasksCount,
        return CheckCompletedTask();
    }
    return false;
}

private bool CheckCompletedTask()
{
    if (currentTask.Count == 0)
    {
        AllTaskCompleted?.Invoke();
    }
    return true;
}
```

Figure 4. Add task code

4. EXPERIMENT

4.1. Experiment 1

The project I designed solved the problem of discovering how to make the random level generator that could generate the mod cause the fire simulation evenly. Sometimes the level wasn't able to generate level random due to this. The fire mod(tiny things causing fire to spread) can't be connected due to this bug. And what we have to do is try to make the generator more stable on each individual level by importing the random level generator into a different kind and more attached to the map.

To improve the stability of the random level generator in a mod-based game and ensure even fire simulation across levels, I tried to test this hypothesis: By importing the random level generator into a more attached and map-based approach, the stability of the generator will improve, and the fire simulation will be evenly distributed across levels.

So I designed this experiment: Ten experienced gamers who have knowledge of mod-based games use computers with necessary software for game development, and internet access for communication. They will run the game a total of 100 times.

All gamers will be given the same task to generate a set of random levels using the current generator. After completing the task, each gamer will record the number of times the bug showed up: fire simulation unevenly spread across the levels. The data collected will be analyzed using descriptive and inferential statistics to determine if there is a significant difference in stability and fire simulation between ten players.

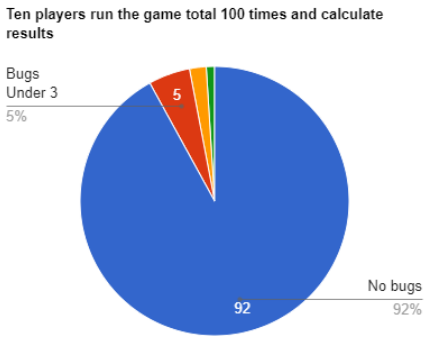

Figure 5. Chart of experiment 1

It works for generating random maps over the landscape, and the fire igniter has been well adjusted around the building. Which it could just still generate a different type of level overall and the fire had been attached to and npc was in the same room. The results provide insights into how to improve the random level generator in mod-based games and ensure even fire simulation.

4.2. Experiment 2

The experiment setting part is scientific in my work is actually the connection of the player to the npc through the process of dialogue. In the tutorial level, the player must accomplish all the tasks required in the task bar to go to the next level. And the npc we're not able to follow directions for me to have a conversation when I collide with him and can't dialoguing over its moving. That increases the difficulty for me to continue based on the dialogue icon remaining on my screen during or after conversation. Which I fix by importing the task to the player. It provides feedback to spot a target in order to trap along an arrow to show position always. And that is much benefit for advancing catching and by process of adding tasks through dialogue.

With the completion of the Experiment 1. Each player will also rate the stability of the generator on a scale of 1 to 9. The data collected will be analyzed using descriptive and inferential statistics to determine if there is a significant difference in stability and fire simulation between the ten players.

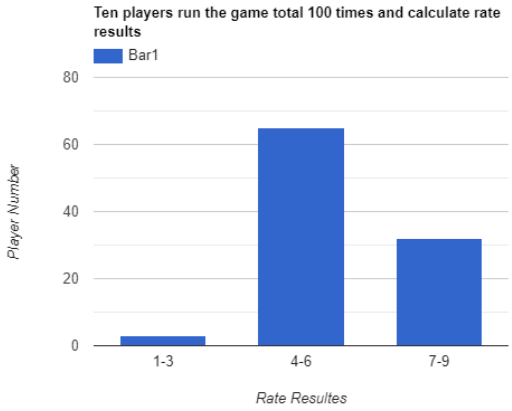

Figure 6. Graph of experiment 2

Based on the provided data, we can analyze the results of the experiment as follows:

The stability rating scale ranges from 1 to 9, with 1 being the lowest and 10 being the highest stability rating. According to the data provided, out of the ten players who participated in the experiment:

Three participants rated the stability of the generator as 1-3. Sixty-five participants rated the stability of the generator as 4-6. Thirty-two participants rated the stability of the generator as 7-9. We can use descriptive statistics to calculate the mean stability rating for all participants.

Mean stability rating = ((3 x 2) + (65 x 5) + (32 x 8)) / (3 + 65 + 32) = 5.53

This indicates that, on average, the players rated the stability of the generator between 4 and 6, which is moderately stable. However, the distribution of the ratings suggests that some players found the generator to be very unstable (rating of 1-3), while others found it to be very stable (rating of 7-9).

Overall, the experiment suggests that the new map-based approach with an attached random level generator led to moderate stability ratings, but there is still room for improvement. Further experiments may be needed to determine the most effective way to improve the stability of the generator and ensure even fire simulation across levels.

The experiment process is challenging for every step built upon every other step. It requires not only your understanding but also participation and logical ideology. It could support you on how to code that you might rely on in your game as well. The experiment results are quite successful after many engagements and failures. It somewhat exceeds my expectations. Which from multiple parts of my game are already advanced. Whether it is the controlling of the player or task system furthering the dialogue system from npc to trigger the next step of keeping the game in processing.

5. RELATED WORK

Unity Linter: A static analysis tool that supports Unity video game developers to detect seven types of bad smells we have identified as relevant in video game development [10]. Such smell types pertain to performance, maintainability and incorrect behavior problems. Our project works differently, mine is more focused on game design, he's more preferred on working on creating

testing tools. Which mine is mostly building a game, and the related work is more helping creating games. There are many beneficial parts for advancing the game. The is a very helpful tool. There are many differences between my project compared to his, in which the development engine and where to publish are totally different, and the intended audience is also different.

Development of RPG-game with built-in adaptive artificial intelligence on the Unity game engine [11]. Objectives of study: development of the system of adaptation of playing situations depending on the actions of the player and attachment of this mechanism to the playing process. The creator's work is quite similar to my project. We have a similar structure by all mentioning the process of how the function works. Playing situations depend on the actions of the player and attachment of this mechanism to the playing process. Which provide strong interesting companies for the players and audiences. RPG games have more freedom on AI control, which we all have RPG controls in game, but the simulation is different on adaptation is also different/ And where design to do different certain things.

The focus is put on using the built-in Unity components in a MOBA setting, developing additional behaviors using Unity's Scripting API for C# and integrating third party components such as the networking engine, 3D models, and particle systems created for use with Unity and available through the Unity Asset Store [12]. This paper discusses implementation of a typical MOBA game prototype for Windows platform in a popular game engine Unity 5. The focus is put on using the built-in Unity components in a MOBA setting. My game focuses more on FPS and TPS, and MOBA is completely a different point of view for benefiting different groups of people. MOBAs have the ultimate goal of destroying the enemy base. Players must work together to acquire items and territory using unique characters, each with their own advantages and weaknesses. And My RPG is more focused on certain objects and complete tasks. And as a change of point of view switch to fps or tps.

6. CONCLUSIONS

The game engine I utilize is Unity, and I use it to create a simulation game all about avoiding disasters made due to human or natural mistakes [14]. I make separate scenes for different effects, there are three(four) scenes in total right now, the beginning scene, tutorial scene, gaming scene and ending scene. The beginning scene shows a variety of effects for build up setting, how to play, and especially the next scene tutorial scene. The setting you are able to change the volume of the game and costume your own keyboard for controlling base on yourself. The how to play shows how to accomplish the level and get to the next level. In the tutorial scene, it gives the player exactly what to play. To accomplish the goal on the board for upgrading. It told you how to change your pov(point of view), how to move and how to talk to npc and what you need to do and giving a fundamental practice for you to be ready to go to the next level. The actual gaming scene is about the topic, escaping through fire disaster. That you must leave the build that is on fire and also talk to teacher npc and student npc. That fire spread fast and caused damage. Which it tells what to do while you are facing difficulties for the actual fire in life. Lastly is the ending scene. I have the opinion to repeat the game and go to the next level that is still under creation.

There are many limitations due to the project I create and also limitations from unity and also myself. The Unity game engine is a well developed platform for video game designers [15]. There are many benefits and disadvantages compared to other game production engines.

The future I would apply more information to the project under creation, find more realistic data and improve on different variety disaster scenes. By adding more common cases and advancing the method of escaping. To be more interesting as well! By learning more and studying more. It can prevent more cases of death from happening by simulating the different dangers you might

have come to meet due to the different disasters. I hope I can be well accessed and really benefit people and help people learn more stuff in this game as they are interested in. Solving these limitations in the future must require more time and more knowledge of unity.

REFERENCES

[1] Maitra, Asit. "School accidents to children: time to act." Emergency Medicine Journal 14.4 (1997): 240-242.
[2] Meng, Die, et al. "Survey and countermeasure discussion of college students' campus fire safety." Procedia Engineering 135 (2016): 25-28.
[3] Yingge, H. E., et al. "Risk state assessment of coal miners' habitual violation behavior." China Safety Science Journal 30.12 (2020): 62.
[4] Helander, Martin G. "Safety hazards and motivation for safe work in the construction industry." International Journal of Industrial Ergonomics 8.3 (1991): 205-223.
[5] Cosentino, Valerio, Javier Luis, and Jordi Cabot. "Findings from GitHub: methods, datasets and limitations." Proceedings of the 13th International Conference on Mining Software Repositories. 2016.
[6] Juliani, Arthur, et al. "Unity: A general platform for intelligent agents." arXiv preprint arXiv:1809.02627 (2018).
[7] Thorn, Alan, and Alan Thorn. "Scripting with C# in Godot: Common Tasks." Moving from Unity to Godot: An In-Depth Handbook to Godot for Unity Users (2020): 53-103.
[8] Rivera, Gabriel, Kenneth Hullett, and Jim Whitehead. "Enemy NPC design patterns in shooter games." Proceedings of the First Workshop on Design Patterns in Games. 2012.
[9] Fox, Edward A., and Sharat Sharan. "A comparison of two methods for soft boolean operator interpretation in information retrieval." (1986).
[10] Borrelli, Antonio, et al. "Detecting video game-specific bad smells in unity projects." Proceedings of the 17th international conference on mining software repositories. 2020.
[11] Ibrahim, Mohd Shukri, et al. "Information technology club management system." Acta Electronica Malaysia 2.2 (2018): 01-05.
[12] Polančec, Domagoj, and Igor Mekterović. "Developing MOBA games using the Unity game engine." 2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). IEEE, 2017.
[13] Cavallo, Eduardo, and Ilan Noy. "Natural disasters and the economy—a survey." International Review of Environmental and Resource Economics 5.1 (2011): 63-102.
[14] Patil, Pratik P., and Ronald Alvares. "Cross-platform application development using unity game engine." Int. J 3.4 (2015).
[15] Haas, John K. "A history of the unity game engine." Diss. Worcester Polytechnic Institute 483.2014 (2014): 484.