



LAUREA
UNIVERSITY OF APPLIED SCIENCES
Together we are stronger

Developing a training tool prototype for IN-ACHUS: the Smart Triage system in a game-based approach using the Blender Game Engine

Çelik, Ali

Laurea University of Applied Sciences

Developing a training tool prototype for INACHUS: the Smart Triage system in a game-based approach using the Blender Game Engine

Ali Çelik
Business Information Technology
Bachelor's Thesis
April, 2017

Çelik, Ali

Developing a training tool prototype for INACHUS: the Smart Triage system in a game-based approach using the Blender Game Engine

Year	2017	Pages	30
------	------	-------	----

This thesis project has been executed for the INACHUS research project. INACHUS is an EU funded research project which aims to optimize the effectiveness of the urban search and rescue teams. The main objective of this thesis project was to implement a training tool prototype to evaluate and prioritize the medical condition of patients using Smart Triage for the INACHUS project as a part of Laurea University of Applied Science's contribution to the research project.

The triage concept is explained and the SALT, START and the Smart Triage systems and their efficiency are discussed in the thesis report. The Smart Triage algorithm is also explained step-by-step. The advantages of the using a game-based approach are also discussed.

The development process that is used for this project was the incremental build model. Meetings with INACHUS representatives were held according to the internal deadlines. The Blender Game Engine, Python programming language, and the Smart Triage algorithm are used for this thesis project's implementation.

As an outcome of the project, a training tool prototype was created. The first part of this training tool is a game prototype system which allows the users to interact with patient objects, observe the patient data inside of the game and let the users predict the treatment priority for each patient and tell them if they were right with their predictions. The second part of the project is a software system which extracts the data into a comma separated values file for each patient and then evaluates the patients' condition based on Smart Triage algorithm and allows the users to observe the patient priority based on the patient data that were created in the game.

The prototype system works successfully and it has passed the software testing session without any sign of error. The INACHUS representatives were highly satisfied with the outcome of the prototype, however, the future development is required for the final training tool. Due to lack of time and resources, this project could not be tested by the urban search and rescue members, thus the effectiveness and the efficiency of this training tool prototype is unknown. Suggestions for the further development regarding the game content, graphical user interface, patient detection, automation of the patient distribution, automation of the game scene, and the importance and the requirement for the opinions of the urban search and rescue members are described in the thesis report.

Keywords: INACHUS, Smart Triage, Blender Game Engine

Table of Contents

1	Background.....	5
1.1	INACHUS and Laurea University of Applied Sciences.....	5
2	Theoretical framework.....	6
2.1	Triage and Smart Triage.....	6
2.2	Game-based approach and learning.....	9
2.3	Blender Game Engine.....	9
2.4	Python.....	9
3	Objectives.....	10
3.1	Limitations.....	10
4	Methodology.....	10
4.1	Software requirements.....	11
5	Implementation.....	13
5.1	Game scene.....	13
5.2	Player object movements.....	13
5.3	Dialogues.....	14
5.4	Patient data.....	14
5.5	Patient detection.....	15
5.6	Player and patient interaction.....	16
5.7	Data extraction.....	17
5.8	Data classification.....	18
5.9	User input collection and validation.....	18
6	Testing the software.....	20
7	Results.....	25
8	Future development.....	26
9	Conclusion.....	27
	References.....	28
	Figures.....	30

1 Background

Catastrophic events and emergency situations are causing an extreme number of casualties per year. According to The Human Cost of Natural Disasters: a global perspective report, more than 1.35 million people lost their lives because of the natural disasters between the years 1994 and 2013 (2015). These unfortunate events are inescapable for the society; however, their damage is reducible with the right preparation techniques and the right tools (Mathbor 2007).

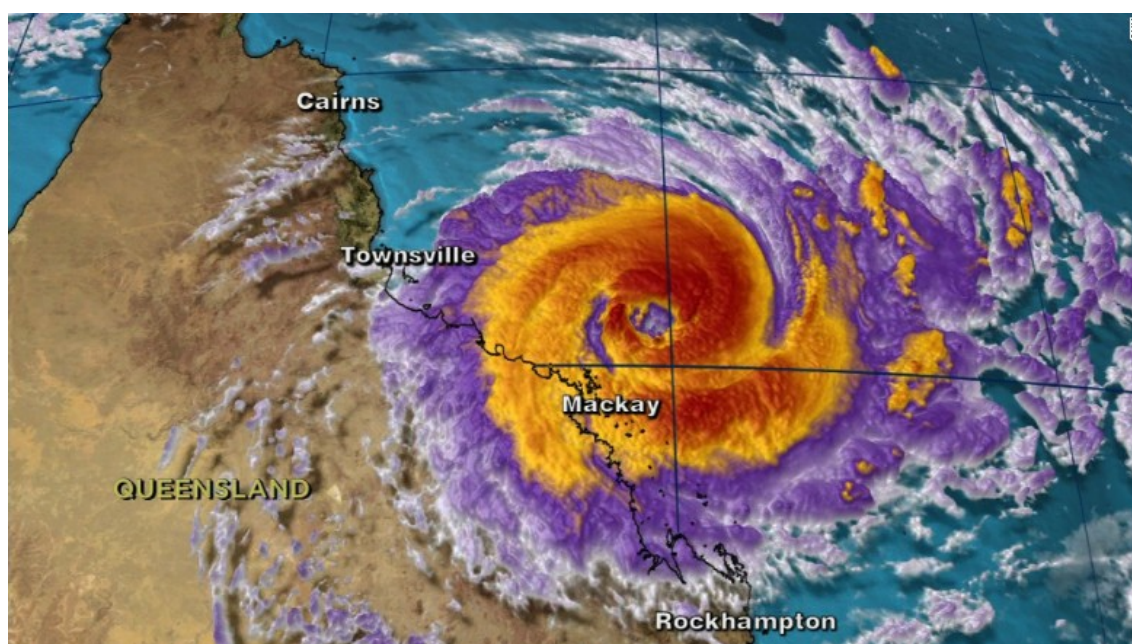


Figure 1: Cyclone Debbie in Queensland Australia, March 2017.
(CNN, 2017)

1.1 INACHUS and Laurea University of Applied Sciences

INACHUS: “Technological and Methodological Solutions for Integrated Wide Area Situation Awareness and Survivor Localization to Support Search and Rescue (USaR) Teams” is a European Union funded research project which aims to improve the effectiveness of the USaR teams and the time efficiency of the rescue operations (INACHUS Press Release 2015).

INACHUS project plans to achieve these aims with twelve work packages;

- “WP1 - Scenarios Definition, User/System Requirements
- WP2 - Framework Design and Interoperability Issues
- WP3 - Simulation Tool for Structural Damage Analysis and Casualty Estimation
- WP4 - Wide-Area Surveillance Tools for monitoring of Collapsed buildings

- WP5 - Victim Localization Solutions
- WP6 - Development of the Emergency Support System
- WP7 - Secure Communications and Positioning Issues
- WP8 - System Integration
- WP9 - Piloting Activities- System Validation
- WP10 - Dissemination, Exploitation and Training Activities
- WP11 - Evaluation and Consideration of Societal Impacts, Legal/Ethical Issues and Standards-Guidelines
- WP12 - Project Management, Quality Assurance and Reporting”
(INACHUS Methodology nod)

Laurea University of Applied Sciences (Laurea UAS) is one of the partners of INACHUS project. Laurea UAS contributes the INACHUS under the two work packages: WP3 and WP11 (LAUREA and INACHUS n.d.). Laurea UAS uses the open-source Blender software to develop the WP3 simulation tool.

WP3, the simulation tool for structural damage analysis and casualty estimation will be used to train USaR teams so that they could have a better understanding of the catastrophic scenes. The simulation consists of two parts; the first part is the structural damage analysis and the second part is the casualty estimation. This thesis project aims to contribute the casualty estimation part of the simulation tool.

The main aim of the casualty estimation part is to predict the approximate location of the victims by simulating catastrophic situations. The secondary aim of the casualty estimation part is to simulate health conditions of the injured people and train the USaR teams to decide the treatment priority by using a triage system. A game-based approach was chosen to apply triage system virtually for this thesis project to keep trainees interested.

2 Theoretical framework

2.1 Triage and Smart Triage

Triage is an injury and illness classification system which prioritize patients based on criticalness of their condition by following certain algorithms (Iserson & Moskop 2007). There are number triage systems exists such as Simple Treatment and Rapid Transport (START), Sort, Assess, Lifesaving interventions, Treat and Transport (SALT), and Smart Triage.

The general approaches of the triage systems are similar. Based on the patient data such as the ability to walk, the ability to breathe, pulse and the ability to obey simple commands, triage systems classifies patients for treatment priority. However, the order of the examination,

label types and different strategies for classification creates the different results for the treatment priorities.

START triage algorithm is developed by the California Fire and Marine Department (REMM 2017). START triage classifies the treatment priority by checking the walking ability, respirations, existence of radial pulse, and ability to obey simple commands. START algorithm has four different classification levels; deceased, immediate, delayed, and minor (Jenkins, McCarthy, Sauer, Green, Stuart, Thomas and Hsu 2007). START triage has been widely used in USA, however, its efficiency was heavily criticized. In 2003, a train accident aftermath showed that the overall accuracy of the START triage was only 44.6% (Kahn, Schultz, Miller and Anderson 2009).

SALT triage algorithm is developed by the US Centers for Disease Control and Prevention. SALT triage consists of two steps; the first step is global sorting. In the global sorting USaR members classifies injured people in three labels these are; assess first, assess second, and assess third. If the injured people can walk, their priority label will automatically become the “assess third”. If the injured people can wave or show any purposeful movement towards the USaR members their label will be the “assess second” and if they cannot move or they are obviously injured in a life-threatening way, their assessment label will become the “assess first”. After this first step, the second step of the SALT algorithm begins, which is the individual assessment. In the individual assessment, injured people will be checked for bleeding, breathing, ability to obey commands and their chance of survival with the current resources of the USaR team members. Based on these data injured people will be classified under four tags; expectant, immediate, minimal and delayed (Cone, Serrra and Kurland 2011). SALT triage has a high accuracy rate with 80+% when compared to START triage algorithm in the same catastrophic events (Lerner, Schwartz, Coule and Pirallo 2010).

Smart Triage is a mass casualty incident triage system developed by TSG Associates in Halifax, Yorkshire, UK (Smart MCI n.d.). In the Smart Triage system, first, USaR team members will check if the injured people can walk, if a person can walk, their treatment priority becomes priority 3. Next, they will check the respirations, if a person does not breathe, they will immediately position the airway and check the respiration again, if the patient still does not breathe, they will tag this person as dead and if the patient starts breathing after positioning the airway, the patient’s treatment priority becomes priority 1 (urgent). If a patient’s respiration is positive in the first check, USaR team members will check patient’s respiration frequency, if the frequency is over 30 times in a minute, the treatment priority of that patient becomes 1. If the respiration frequency is under 30 times in a minute, USaR team members will check the patient’s capillary refill time, if it takes more than 2 seconds, USaR team members will look for signs of bleeding and then will tag the patient as priority 1. If the capillary

refill time takes less than 2 seconds, USaR team members will check if the patient can obey simple commands, if the patient cannot obey simple commands, USaR team members will look for signs of bleeding and then will tag the patient as priority 1. If the patient can obey simple commands, the priority of that patient will become priority 2. A virtual reality experiment for Smart Triage and SALT comparison showed that treatment priority classification with Smart Triage was much more fast and accurate than the SALT system (Cone et al. 2011).

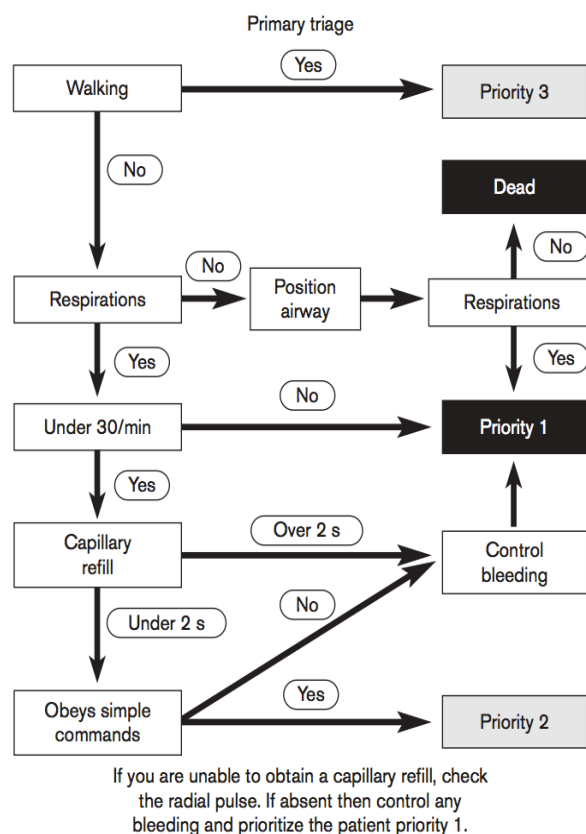


Figure 2: Smart Triage algorithm.

(Cone et al. 2011)

Due to its high accuracy comparing the other popular triage systems, Smart Triage system was chosen for this thesis project. The classification criteria that is used in the Smart Triage algorithm such as walking, respirations, respiration frequency, capillary refill and ability to obey simple commands used as patient data in this thesis project to allow the software to perform the Smart Triage algorithm and show users to priority assigning based on the patient data.

The main aim of implementing this triage system into training prototype was to allow USaR members to practice Smart Triage's classification system. However, the "position airway" step in the Smart Triage algorithm was an intervention method rather than classification.

Thus, in this training tool prototype, positioning the airway was not calculated, if an injured person's respiration is negative, their priority will automatically become priority 1.

2.2 Game-based approach and learning

The popularity and the availability of the video games are constantly increasing over time. Different genres and different game modes exist in today's world and each of these different genres and modes targeting people with different characteristics, ages, and genders (McGonigal 2011, 20-21).

Video games are used in education since the early 1980s. Some of the most popular video games of that era selected by some of the educators and used in the classrooms. Video games such as Pac-Man and Super Mario Brothers 2 showed a clear structure to educators about key points of the game design essentials and how they could be adapted for educational purposes (Squire 2003).

Game-based approach has successfully used for educational purposes for the health field. In 2007, 33 surgeons participated a program showed that surgeons who played video games more than 3 hours a week put a better performance in the Rosser Top Gun Laparoscopic Skills and Suturing Program when compared to their non-gamer colleagues (Rosser, Lynch, Cuddihy, Gentile, Klonsky and Merrel 2007).

Additionally, a game based training simulation called Sidh was used for firefighters' training in Sweden and collected positive feedbacks from firefighters regarding the game's ability to teach learning objectives (Backlund, Engström, Hammar, Johannesson and Lebram 2007).

2.3 Blender Game Engine

Blender Game Engine (BGE) is a part of the Blender, an open-source 3D computer graphics production software (About Blender n.d.). The main purpose of the BGE is to create game-based environments for real-time projects (Game Engine n.d.).

Blender Game Engine was used for this project to have a better collaboration with Laurea UAS' simulation tool.

2.4 Python

Python is a high-level multi-paradigm-type programming language (Rossum 2007). Python is used for this project due to BGE's support to Python language (Game Engine n.d.).

3 Objectives

The main objective of this project is to develop a training tool prototype to evaluate and prioritize the medical condition of patients using Smart Triage system in a virtual environment which is created with using Blender Game Engine and get the user input regarding the prioritizing order and compare it with the software's result and show the users the accuracy of their prediction.

The secondary objective is to adapt this virtual environment to a maze-like structure to illustrate three-dimensional structures in two-dimension.

3.1 Limitations

Requirements for this project were set according to available resources. These resources were not enough to create a complete training tool or develop an advanced training tool prototype; however, they were enough to complete the first version of the prototype. Some approaches regarding technical complexity were not the most optimal ones. Additionally, analyzing the software outcome and its efficiency were not possible. Thus, overall quality and effectiveness of the software system is debatable.

4 Methodology

Incremental build model is used as software development method to gain a clear understanding of the BGE and other technical requirements of the project. Meetings with project representatives held according to the internal deadlines.

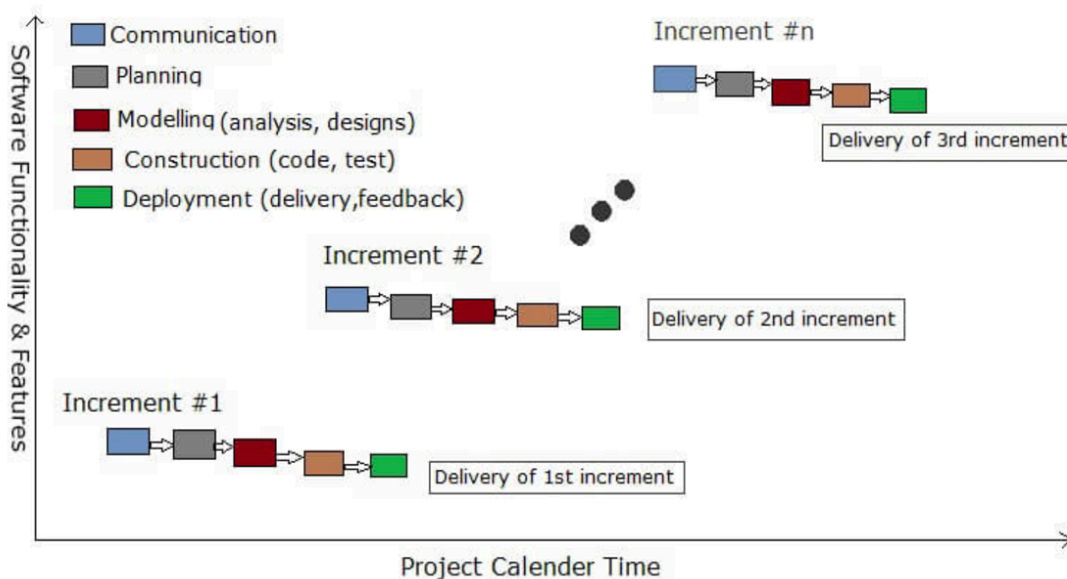


Figure 3: Incremental build model
(Tilloo 2016)

The first increment of the project was to build the game scene and once this was done the next increments were with order; implementing player movements, creating dialogues, creating patient data, implementing a patient detection system, enabling player-patient interaction, extracting the data, classifying the data and finally getting the user input for classification predictions and analyzing the accuracy of their prediction. Each of these increments had their own internal deadlines.

The project required familiarity with the BGE's user interface and its Python programming language support as well as the Smart Triage algorithm.

4.1 Software requirements

Software requirements of the project determined by analyzing the Smart Triage algorithm and discussing the potential project requirements with the INACHUS representatives.

Since the software implementation method was the incremental build model, increments mentioned above formed the overall software requirements.

Requirements for the game scene were the building a maze-like structure and creating one player object and three injured person objects for the game part of the software. Injured person objects would later carry certain data to share with the player object for triage training purposes.

Requirements for the player object movements were to implement a keyboard directed movement system for the player object so that during the software usage players would be able to walk around the maze and search for injured person objects. W, A, S, D keys were chosen for player object movements due to their popularity in the gaming industry (Wilde 2016).

Dialogue requirements were to implement a small dialogue between player and injured person objects and show the Smart Triage related data to the player so that they would know which type of data are processing by the software.

Patient data requirement was to build an algorithm that would produce new patient data in each game session. Patient data types are adopted from Smart Triage algorithm.

The requirement for the patient detection was to build a patient detection system in the maze-like structure so that player would have an idea where to look for the injured person object. In real life, there are number of detection systems exist to detect people in collapsed structures. However, in this thesis project, an arrow shaped detection system which would work like a compass decided as the detection system to cover different types of detection strategies under the one feature.

Player-patient interaction requirement was to start to dialogue between player and patient objects when there are no obstacles between objects and the distance is reasonable. The return key has chosen for the starting the interaction between the player and the patient objects. Space key has chosen to continue the dialogue and complete the interaction at the end of the dialogue.

The requirement for data extraction was to extract the patient data which is created inside of the game part of the software system in a spreadsheet file format so that after the game session users would have a chance to see the patient data and their treatment priority level. Comma-Separated Values (CSV) file format chosen due to its popularity as a data file format (Shafranovich 2005).

Data classification requirements were to classify patient priority based on Smart Triage algorithm and show this classification in the extracted data file.

The requirement for the user input collection and validation was to collect the user's prediction regarding the treatment prioritization via keyboard input and compare their prediction with the computer's result and show them about the accuracy of their guesses inside of the game.

5 Implementation

5.1 Game scene

Game scene was created in a maze-like structure to illustrate three-dimensional structures in two-dimension. The scene includes one floor, number of walls, three injured person objects, and one player object.

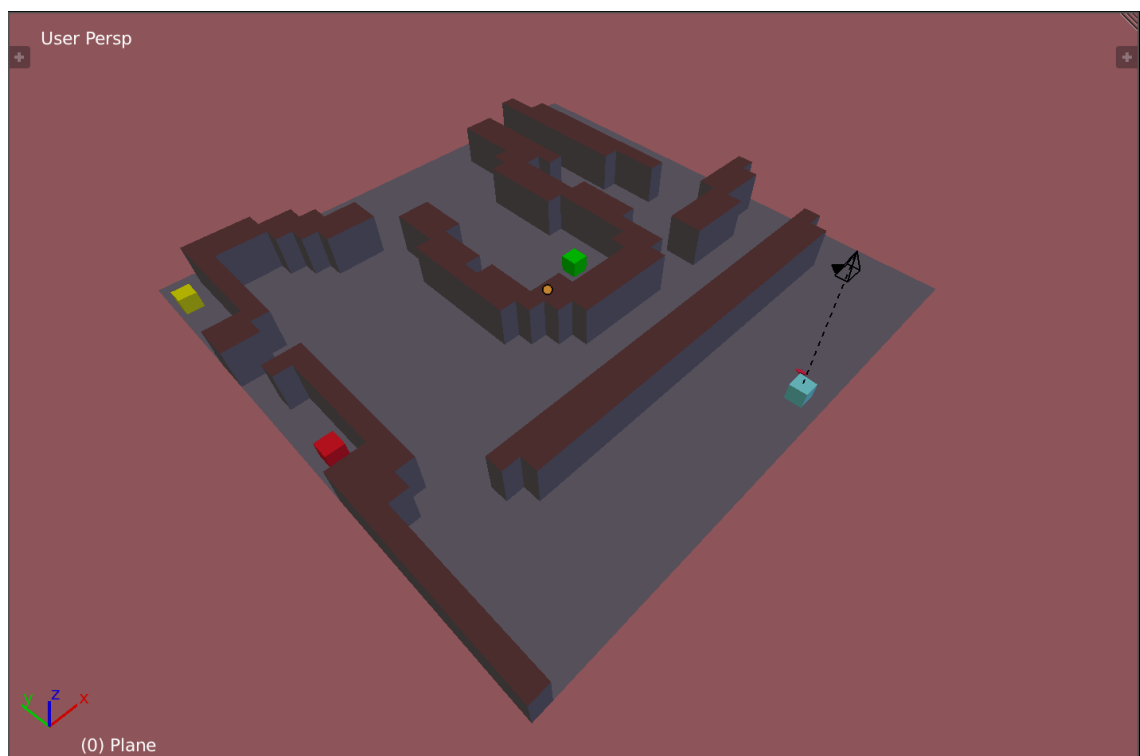


Figure 4: Game Scene

5.2 Player object movements

The player object can be moved by pressing W, S, A, D key from the keyboard. W and S keys used for moving forward and backward, A and D keys are used for moving left and right. Movements are implemented by using Blender's own Python library BGE.

```

cont = bge.logic.getCurrentController()
own = cont.owner
keyboard = bge.logic.keyboard

wKey= bge.logic.KX_INPUT_ACTIVE == keyboard.events[bge.events.WKEY]
sKey= bge.logic.KX_INPUT_ACTIVE == keyboard.events[bge.events.SKEY]
aKey= bge.logic.KX_INPUT_ACTIVE == keyboard.events[bge.events.AKEY]
dKey= bge.logic.KX_INPUT_ACTIVE == keyboard.events[bge.events.DKEY]

|
if wKey:
    own.applyMovement([0,0.2,0],True)
if sKey:
    own.applyMovement([0,-0.2,0],True)
if aKey:
    own.applyMovement([-0.2,0,0],True)
if dKey:
    own.applyMovement([0.2,0,0],True)

```

Figure 5: Player movements with Python

5.3 Dialogues

In game dialogues implemented to illustrate a sample dialogue between player and injured person objects and show the player the patient data of the injured person object. Dialogues include a small introductory conversation between player and injured person objects and then shows the patient data to the player.

5.4 Patient data

Patient data consist of the ability of the walking, respirations, respirations per minute, capillary refill, and ability to obey simple commands. The data types and their values are based on Smart Triage algorithm.

The text data and dialogue files for all the patients were created at random in the beginning of each game session by using Python's Random library. The algorithm works in a way that it will run only once in each session with overriding the previous session's data files. If there are no previous session's data files exists, the software will create them.

```

while i < 3:
    with open(fileDirs[i], "w") as f:
        yesNo = ['Yes', 'No']
        aboveUnder = ['Above 30', 'Under 30']
        overUnder = ['Over 2 s', 'Under 2 s']
        greetings = ['Hey! Help is here!', 'Hey! I am going to help you!', 'Do not worry, I am going to help you.']
        decisions = [random.choice(yesNo), random.choice(yesNo), random.choice(aboveUnder), random.choice(overUnder), random.choice(yesNo)]
        f.write(random.choice(greetings) + "\n")
        f.write("Now I am going to examine you.\n")
        f.write("Walking: " + decisions[0] + "\n")
        if decisions[0] == "Yes":
            f.write("Respirations: Yes\n")
        else:
            f.write("Respirations: " + decisions[1] + "\n")
        if decisions[1] == "No":
            f.write("Respirations per min: - \n")
        else:
            f.write("Respirations per min: " + decisions[2] + "\n")
        f.write("Capillary refill: " + decisions[3] + "\n")
        if decisions[1] == "No":
            f.write("Obey simple commands: - \n")
        else:
            f.write("Obey simple commands: " + decisions[4] + "\n")
        f.write("Okay, I am going to help you.")
        with open(dataDirs[i], "w") as d:
            d.write("Walking: " + decisions[0] + "\n")
            if decisions[0] == "Yes":
                d.write("Respirations: Yes\n")
            else:
                d.write("Respirations: " + decisions[1] + "\n")
            if decisions[1] == "No":
                d.write("Respirations per min: - \n")
            else:
                d.write("Respirations per min: " + decisions[2] + "\n")
            d.write("Capillary refill: " + decisions[3] + "\n")
            if decisions[1] == "Yes":
                d.write("Obey simple commands: " + decisions[4] + "\n")
            else:
                d.write("Obey simple commands: - \n")

```

Figure 6: Random data and dialogue algorithm

5.5 Patient detection

Based on a discussion with the project representatives, an arrow object which is detecting patients in the maze based on their distance to the player object, was implemented to cover different types of detection techniques. The arrow object detects patients by using Blender's Near sensor and Track To actuator with the help of a Python file called trackNearest.py. First, the Near sensor detects the patient objects based on their distances to the player, then trackNearest.py activates and calls the Track To actuator and finally the arrow object changes its direction towards detected patient and shows the player the direction of the patient object.

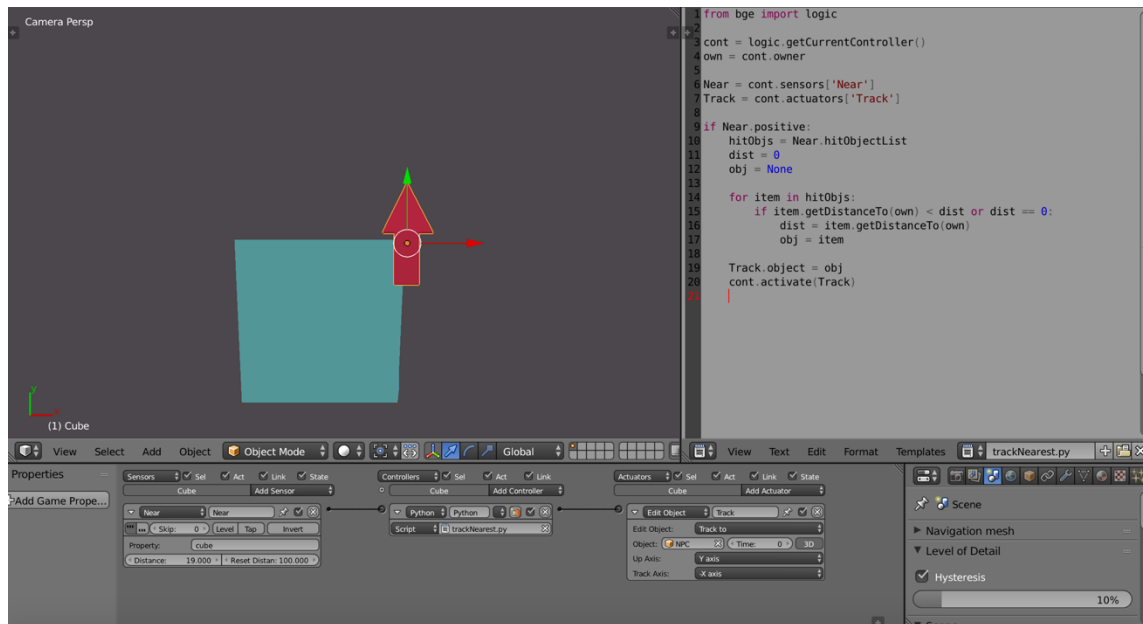


Figure 7: The arrow object, Near sensor, trackNearest.py and Track To actuator

The arrow object is a child object of the player object, which means it will always stick to the player object.

5.6 Player and patient interaction

Player and patient interaction occur when the player encounters with a patient object. By using Blender's Ray sensor, a small red line occurs if there are no obstacles exists between patient. This red line represents the vision, so in other words, when the red line occurs player can see the injured people. After player sees the injured person, an interact sign is shown to the player to tell that the dialogue and the observation can start now.

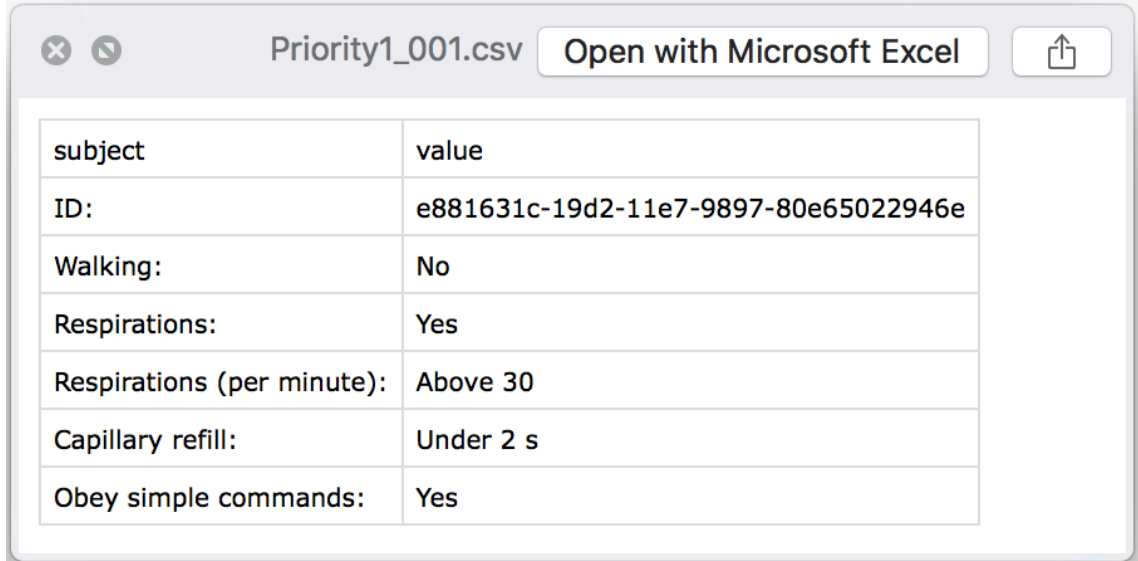


Figure 8: Interact sign and red ray between player and patient objects

After interact sign comes up, the player can start the dialogue by pressing the return key and continue the dialogue by pressing the space key. Once the dialogue is over, the player can look for other patient objects.

5.7 Data extraction

After data values are created and displayed to the player, the software creates a Comma-Separated Values file and extract the data with the help of Python's CSV library. During the extraction process, by using UUID library functionality of Python, a unique ID was created for each injured person and added to the CSV file.



subject	value
ID:	e881631c-19d2-11e7-9897-80e65022946e
Walking:	No
Respirations:	Yes
Respirations (per minute):	Above 30
Capillary refill:	Under 2 s
Obey simple commands:	Yes

Figure 9: Sample CSV file

5.8 Data classification

Data classification process begins after the extraction process. Based on the Smart Triage algorithm and the random values are created by the Python, the software classifies the injured person data and decides the priority. Based on the values, software changes the CSV file names according to their priority. The format of the file name is; “Priority Order”_“Object number”.csv.

```

if walking == " Yes":
    os.rename(path+"injuredData.csv", path+"Priority3_{}.csv".format("001"))
elif walking == " No":
    if respirations == " Yes":
        if respirationsPerMin == " Under 30":
            if capillaryRefill == " Under 2 s":
                if simpleCommands == " Yes":
                    os.rename(path+"injuredData.csv", path+"Priority2_{}.csv".format("001"))
                else:
                    os.rename(path+"injuredData.csv", path+"Priority1_{}.csv".format("001"))
            else:
                os.rename(path+"injuredData.csv", path+"Priority1_{}.csv".format("001"))
        else:
            os.rename(path+"injuredData.csv", path+"Priority1_{}.csv".format("001"))
    else:
        os.rename(path+"injuredData.csv", path+"Priority1_{}.csv".format("001"))

```

Figure 10: Classification process with Python

5.9 User input collection and validation

After the software’s classification process, user input collection and validation begins. The software will ask the user about their prediction of a patient’s treatment priority and will

compare the user's prediction with its own classification. The users can give their prediction by pressing the 1, 2 and 3 keys from the keyboard. Each key represents a priority. If the user is right with their prediction, the game will show the user a "Correct!" sign and if the user is wrong, the game will show a "Wrong!" sign. The user input collection was implemented via BGE's keyboard sensor, message sensor and message actuator. The validation was implemented via Python by analyzing the user's keyboard input and comparing the software's own classification result.

```

priority = "0"
path11 = Path(path+"Priority1_001_temp.csv")
path21 = Path(path+"Priority1_002_temp.csv")
path31 = Path(path+"Priority1_003_temp.csv")

if path11.is_file():
    priority = "1"
    if path11.is_file():
        os.remove(path+"Priority1_001_temp.csv")
if path21.is_file():
    priority = "1"
    if path21.is_file():
        os.remove(path+"Priority1_002_temp.csv")
if path31.is_file():
    priority = "1"
    if path31.is_file():
        os.remove(path+"Priority1_003_temp.csv")

key1 = cont.sensors['1']
if key1.positive:
    own.text = "1 Works"
    if str(key1) == priority:
        own.text = "Correct!"
        priority = "0"
        t = Timer(0.5, change)
        t.start()
    else:
        own.text = "Wrong!"
        t = Timer(0.5, change)
        t.start()
        print(key1)
        print(type(key1))
        print(priority)
        print(type(priority))

```

Figure 11: User input validation

6 Testing the software

After the implementation, software system was tested to observe the overall performance and the accuracy of the prototype.

As the first step of testing, a game session started and the patients were detected in the maze by the guidance of the arrow object;

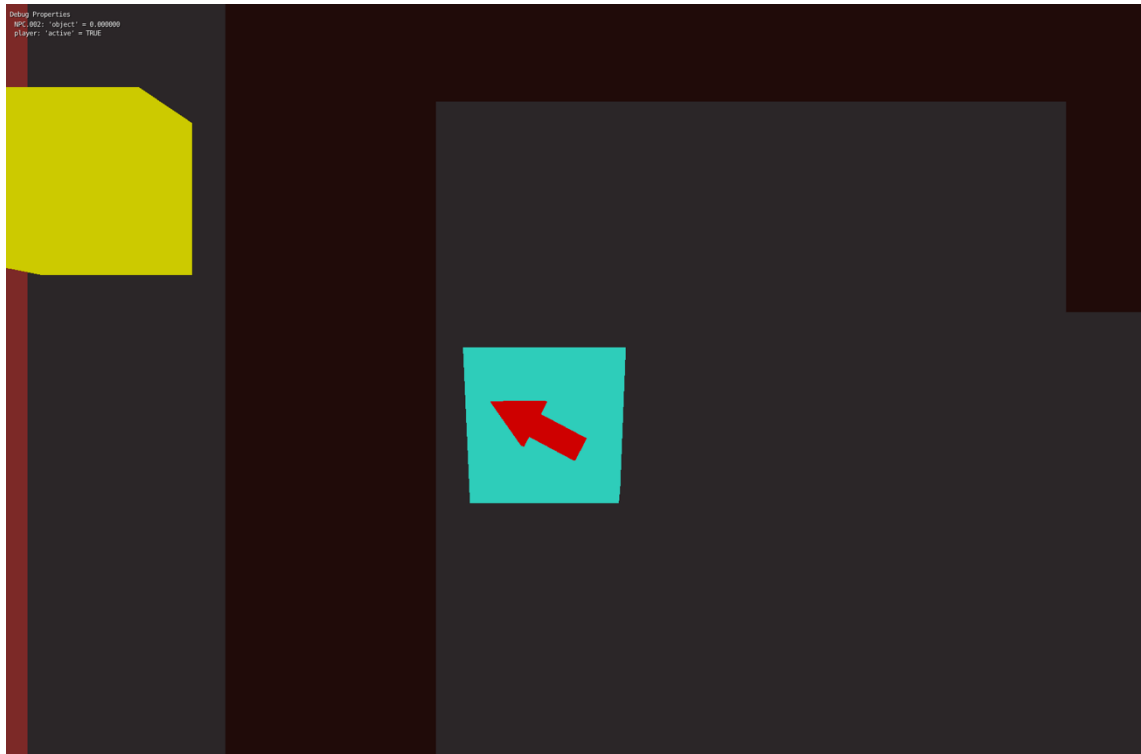


Figure 12: Detecting a patient with the arrow object

After detecting the patients successfully, the player-patient interaction was tested and in-game version of the dialogue data observed for each patient;

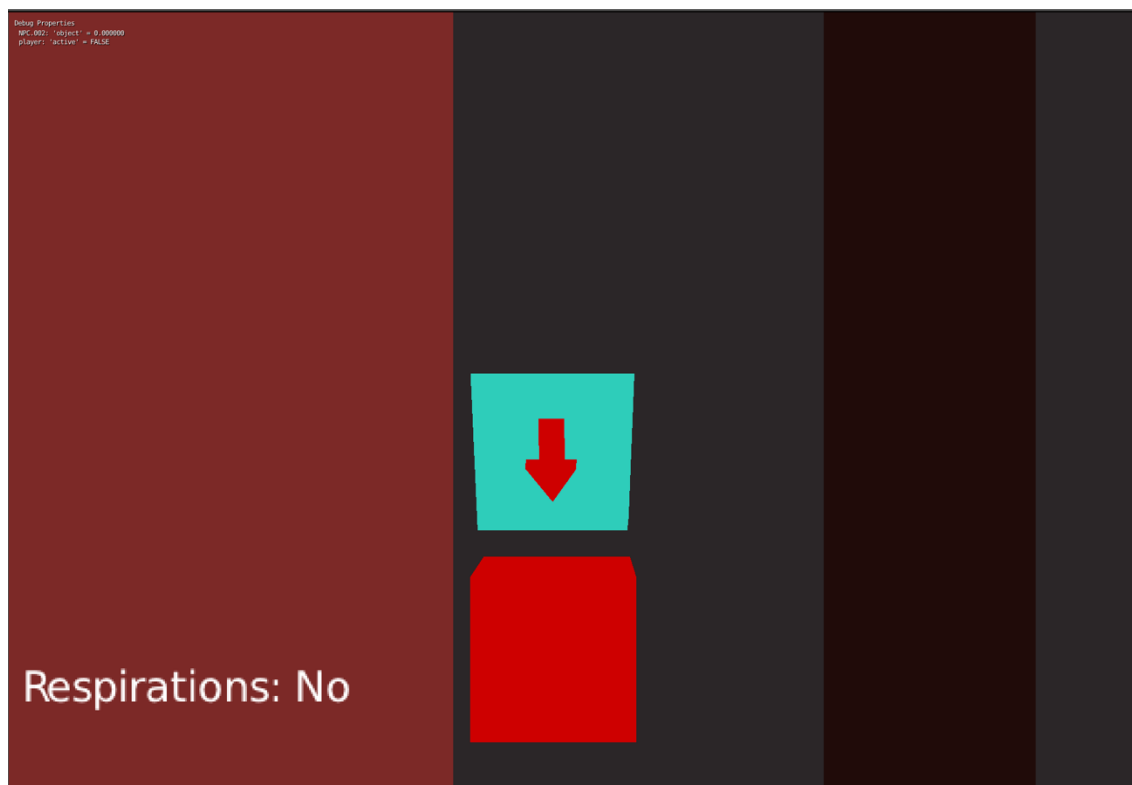


Figure 13: The dialogue scene

After the successful interaction and the in-game dialogues, the text version of the dialogue data was observed for each patient. The in-game dialogue and text dialogue data were matching successfully for each case;

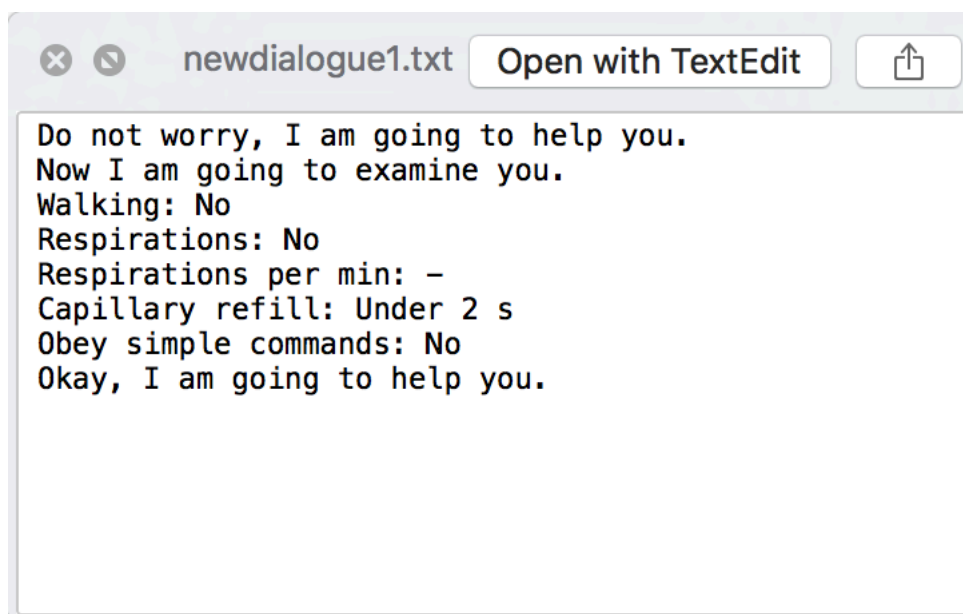


Figure 14: The dialogue data of the first patient object

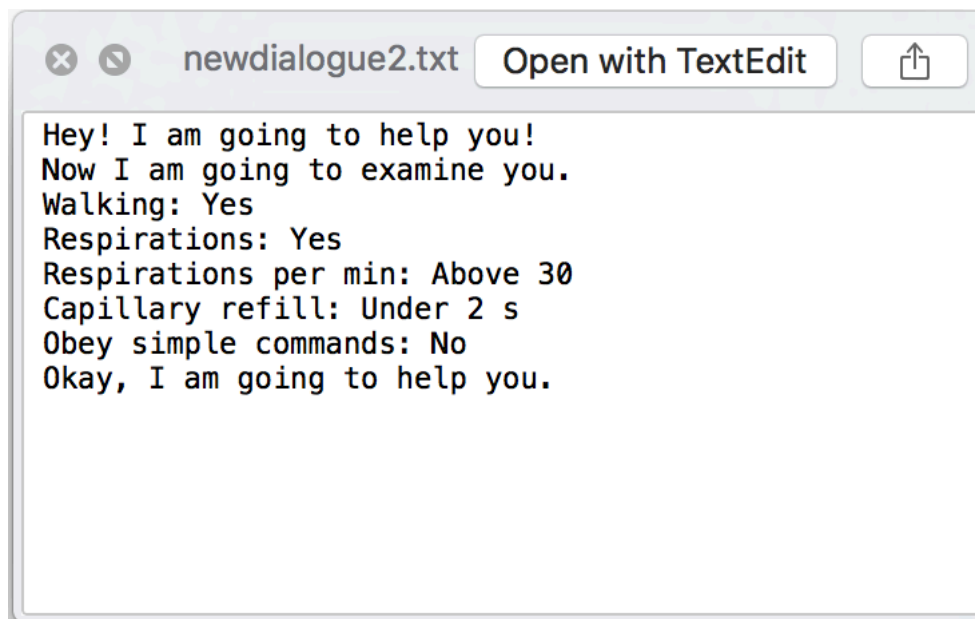


Figure 15: The dialogue data of the second patient object

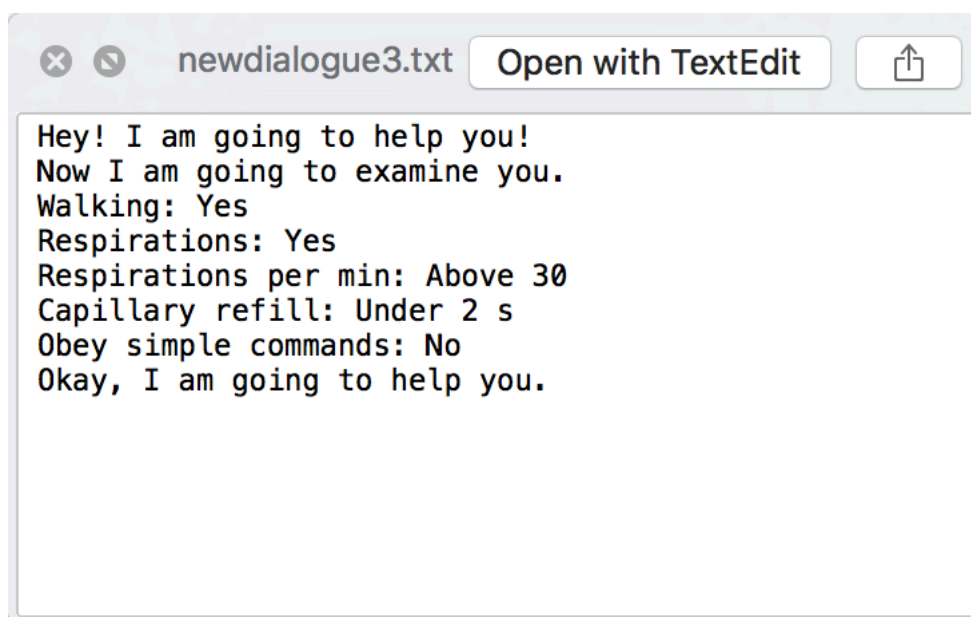
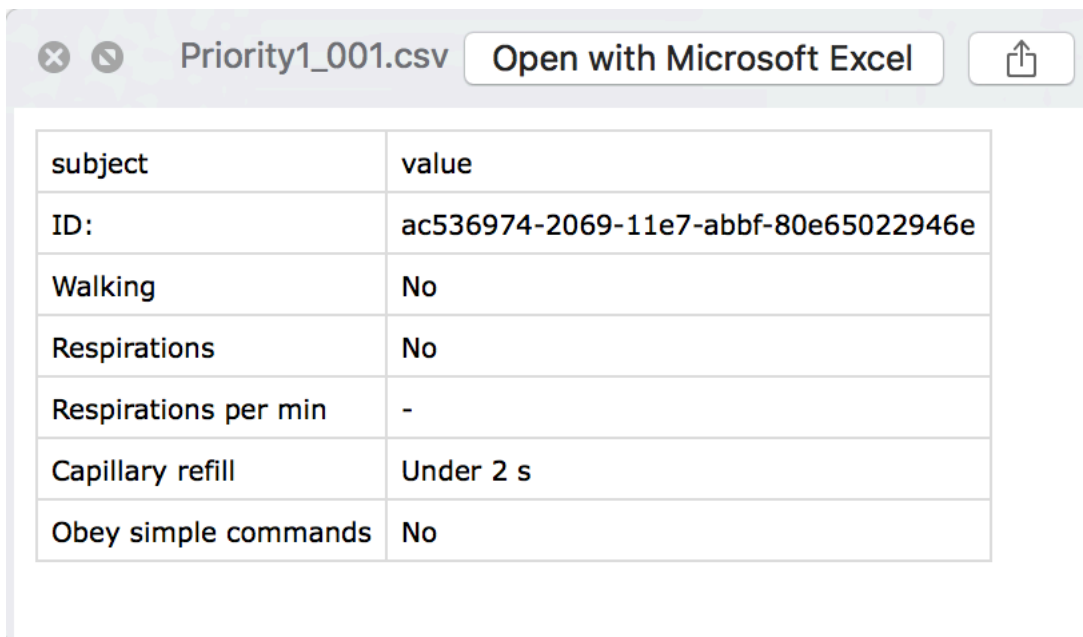


Figure 16: The dialogue data of the third patient object

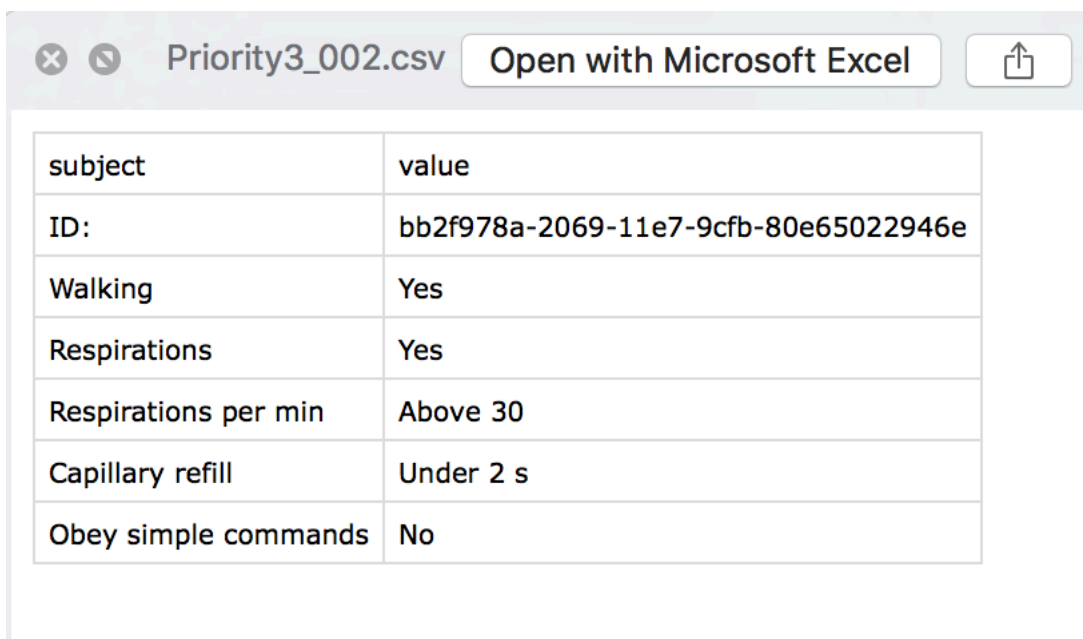
Then the patient text data and CSV data were compared to each other to see if the data were matching. No errors observed.

And then, the CSV data extracted and classified by using the Smart Triage algorithm;



subject	value
ID:	ac536974-2069-11e7-abbf-80e65022946e
Walking	No
Respirations	No
Respirations per min	-
Capillary refill	Under 2 s
Obey simple commands	No

Figure 17: The data file of the first patient with priority classification



subject	value
ID:	bb2f978a-2069-11e7-9cfb-80e65022946e
Walking	Yes
Respirations	Yes
Respirations per min	Above 30
Capillary refill	Under 2 s
Obey simple commands	No

Figure 18: The data file of the second patient with priority classification

subject	value
ID:	b4e24554-2069-11e7-a975-80e65022946e
Walking	Yes
Respirations	Yes
Respirations per min	Above 30
Capillary refill	Under 2 s
Obey simple commands	No

Figure 19: The data file of the third patient with priority classification

All the classifications of the test session were correct. Classifications of the cases were done accurately and CSV file names were changed according to them. In addition, unique ID's were successfully created for each CSV file.

After the data classification, user input collection and validation started;

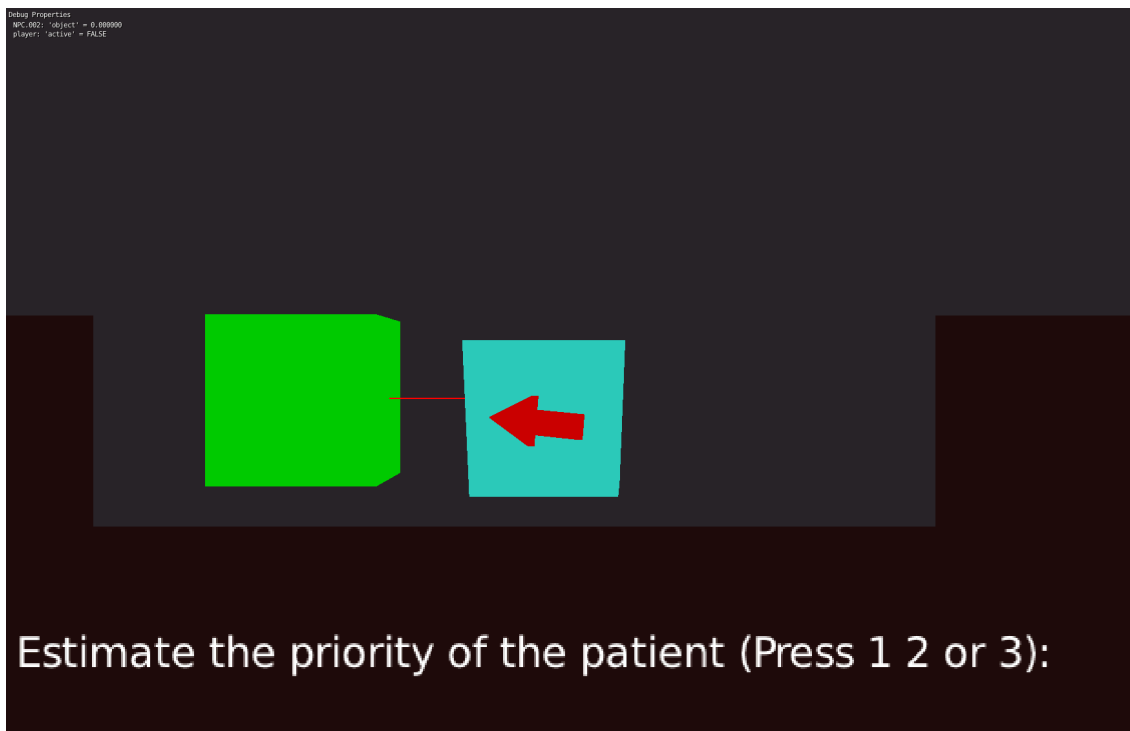


Figure 20: Collecting the user input

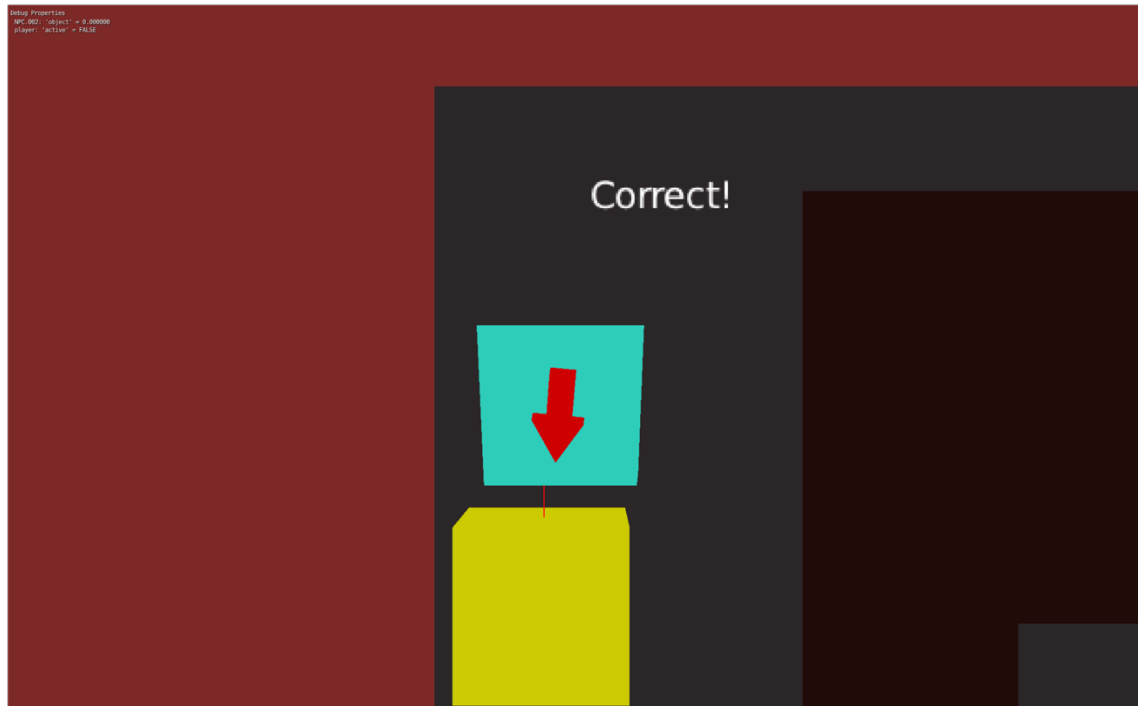


Figure 21: Validation of the prediction

7 Results

At the end of the project, a training tool prototype was created. The structure of the prototype system is the following; the player enters the game, dialogues and patient data are created automatically, based on that data the software extracts the data and creates CSV files and then the software classifies patient priority based on the Smart Triage algorithm changes the name of the CSV files according to their priorities so that the player can see the data and the priority of patients, then software asks users about their prediction regarding treatment priority of patient objects and then compares users' result with its own result and show the users if they were right or not.

The testing completed successfully with a complete accuracy and without any signs of error based on the Smart Triage algorithm.

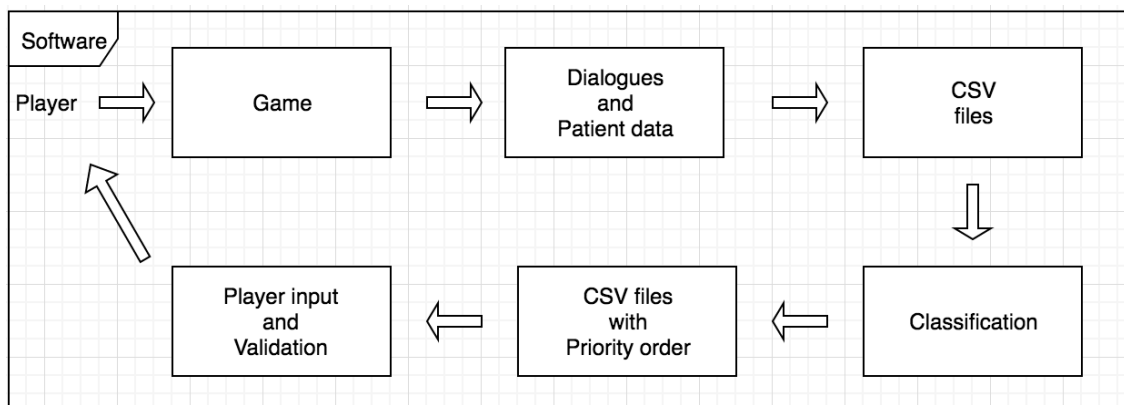


Figure 22: Software system

8 Future development

Future development is necessary to complete the prototype. Game scene, user interface, game content, and the patient detection system needs further development, new systems such as automation of the patient distribution and automation of the game scene can be implemented and getting opinions of USaR team members regarding the software system and its efficiency can be used to evaluate and develop the overall approach of the software system.

The game scene needs more detailed content and different types of maze structures to increase the diversity of the scenarios. Three-dimensional approach can be applied to the game scene instead of two-dimensional approach for better user experience.

The user interface needs better graphical representations of the real-life objects and the people. Realistic approaches regarding the user interface can increase the user experience and give USaR team members a better understanding of the catastrophic scenes in real life.

The game content can be improved by adding more features to the game. For example, a level-based system where each level includes more difficult scenarios for the player can be implemented. In this way, players can encounter with different types of scenarios this could improve the user experience and satisfaction. In addition, a score system can also be implemented. The current version of the game does not have any evaluation system other than the telling the users if they were right with their priority predictions or not. With the implementation of a scoring system, users can collect points for each correct prediction and after every game session, a high score table can be shown to keep the player interested and possibly create a competition feeling.

The patient detection system can be improved by implementing different detection choices. The current version of the software has only one type of detection system. Implementing different types of patient detection systems realistically can improve the overall user experience.

Automation of the patient distribution can be an important development to increase the quality of the software system. In the current version, there are only three patients and their locations are fixed. With the automation of the patient distribution, game diversity can be increased and different scenarios for users can be created.

Automation of the game scene can also be an important development for the software system. Using artificial intelligence techniques such as artificial neural networks or Qlearning, a smart level system can be implemented to the game. In this way, every level would have a unique content based on each player's weaknesses and strengths. In this way, players can observe and improve their theoretical knowledge regarding the triage and rescuing subjects.

Opinions of the USaR team members can be used to evaluate and improve the software system. The current version has not been tested by the professionals therefore, its effectiveness in training USaR members is unknown. Letting rescue professionals to use the software and collecting their feedback regarding its efficiency can be crucial for the final version of the training tool.

9 Conclusion

In conclusion, a training tool prototype was created. Due to project's complexity and time and resource limitations prototype could not be perfected and the future development is required for stability and reliability. However, INACHUS representatives were satisfied with the overall prototype. The current version of the training tool prototype executes Smart Triage algorithm and classifies the patient objects without any failure. The software extracts the data and stores it in CSV format and compares the computer's classification with user's prediction. All the software requirements were executed successfully.

References

About Blender. No date. Introduction. Accessed 2 April 2017. https://docs.blender.org/manual/ko/dev/getting_started/about/introduction.html

Backlund, P. Engström, H. Hammar, C. Johannesson, M. & Lebram, M. 2007. Sidh - a Game Based Firefighter Training Simulation. In: 11th International Conference of IEEE July 2007. Information Visualization. Accessed 21 April 2017. <http://ieeexplore.ieee.org/abstract/document/4272085/>

CNN. Cyclone Debbie prompts evacuations. 2017. Accessed 1 April 2017. <http://edition.cnn.com/videos/weather/2017/03/27/australia-queensland-cyclone-debbie-latest-gray-sot-cnni.cnn>

Cone, D., Serrra, J. & Kurland, L. 2011. Comparison of the SALT and Smart triage systems using a virtual reality simulator with paramedic students. *European Journal of Emergency Medicine*, 18(6), 3.

Game Engine. No date. Introduction. Accessed 2 April 2017. https://docs.blender.org/manual/ko/dev/game_engine/introduction.html

INACHUS Methodology. No date. Accessed 2 April 2017. <https://www.inachus.eu/methodology>

INACHUS Press Release. 2015. Accessed 1 April 2017. <https://drive.google.com/file/d/0BzgAgXQ5LS6lYmFjYtImQjRDcXM/view>

Iserson K. & Moskop, J. 2007. Triage in Medicine, Part I: Concept, History, and Types. *Annals of Emergency Medicine*, 49 (3), 275-276.

Jenkins, J. McCarthy, M. Sauer, L. Green, G. Stuart, S. Thomas T. & Hsu, E. 2007. Mass-Casualty Triage: Time for an Evidence- Based Approach. *Prehospital and disaster medicine*, 23(01), 3-8.

Kahn, C., Schultz, C., Miller, K. & Anderson, C. 2009. Does START triage work? An outcomes assessment after a disaster. *Annals of emergency medicine*, 54(3), 424-430.

LAUREA and INACHUS. No date. About. Accessed 13 April 2017. <https://inachuslaurea.wordpress.com/about/>

Lerner, B., Schwartz, B., Coule, L. and Pirallo, G. 2010. Use of SALT triage in a simulated mass-casualty incident. *Prehospital emergency care*, 14(1), 21-25.

Mathbor, G. 2007. Enhancement of community preparedness for natural disasters. *International Social Work*, 50 (3), 358.

McGonigal, J. 2011. *Reality is broken: Why games make us better and how they can change the world*. New York: Penguin Press.

REMM. 2017. START Adult Triage Algorithm. Accessed on 27 April 2017. <https://www.remm.nlm.gov/startadult.htm>

Rosser, J., Lynch, P., Cuddihy, L., Gentile, D., Klonsky, J. & Merrel, R. 2007. The Impact of Video Games on Training Surgeons in the 21st Century. *Archives of surgery*, 142 (2), 181-182.

Rossum, G. 2007. Python Programming Language. In: USENIX Annual Technical Conference June 2007. Accessed 21 April 2017. http://colenak.ptkpt.net/_lain.php?_lain=3721

Shafranovich, Y. 2005. Common Format and MIME Type for Comma-Separated Values (CSV) Files. Accessed 13 April 2017. <https://www.ietf.org/rfc/rfc4180.txt>

Smart MCI. No date. About Us. Accessed 2 April 2017. http://www.smart-mci.com/about_us.php

Squire, K. 2003. Video games in education. International journal of intelligent simulations and gaming, 2003 (2), 49-62.

The Human Cost of Natural Disasters: A global perspective. 2015. Centre for Research on the Epidemiology of Disasters CRED, 7. Accessed 1 April 2017. http://reliefweb.int/sites/reliefweb.int/files/resources/PAND_report.pdf

Tilloo, R. 2016. What Is Incremental Model In Software Engineering? Accessed 2 April 2017. <http://www.technotrice.com/incremental-model-in-software-engineering/>

Wilde, T. 2016. How WASD became the standard PC control scheme. Accessed 13 April 2017. <http://www.pcgamer.com/how-wasd-became-the-standard-pc-control-scheme/>

Figures

Figure 1: Cyclone Debbie in Queensland Australia, March 2017	5
Figure 2: Smart Triage algorithm.....	8
Figure 3: Incremental build model	11
Figure 4: Game Scene	13
Figure 5: Player movements with Python	14
Figure 6: Random data and dialogue algorithm	15
Figure 7: The arrow object, Near sensor, trackNearest.py and Track To actuator	16
Figure 8: Interact sign and red ray between player and patient objects	17
Figure 9: Sample CSV file	18
Figure 10: Classification process with Python	18
Figure 11: User input validation	19
Figure 12: Detecting a patient with the arrow object	20
Figure 13: The dialogue scene	21
Figure 14: The dialogue data of the first patient object	21
Figure 15: The dialogue data of the second patient object.....	22
Figure 16: The dialogue data of the third patient object	22
Figure 17: The data file of the first patient with priority classification	23
Figure 18: The data file of the second patient with priority classification.....	23
Figure 19: The data file of the third patient with priority classification	24
Figure 20: Collecting the user input	24
Figure 21: Validation of the prediction.....	25
Figure 22: Software system	26