Journal of Theoretical and Applied Information Technology
15th May 2023. Vol.101. No 9
© 2023 Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

DESIGN ADVENTURE ROLE PLAYING GAME WITH PROCEDURAL CONTENT GENERATION USING PERLIN NOISE ALGORITHM

RAUL ANDRIAN[1] , ANGGA ADITYA PERMANA[2*] , ADHI KUSNADI[3]

[1, 2*, 3]Department of Informatic, Faculty of Engineering & Informatics, University of Multimedia Nusantara, Indonesia

E-mail: [1]raul.andrian@student.umn.ac.id, [2*]angga.permana@umn.ac.id, [3]adhi.kusnadi@umn.ac.id

ABSTRACT

Producing quality video game content is not easy. In the process of creating game content, it will require a lot of resources, a long time and expensive costs. This problem can be solved by applying the Procedural Content Generation or PCG method to create game content using certain algorithms without having to create it manually. In this study, the Perlin Noise algorithm was used to create a map for the game that was built. Perlin Noise created realistic visuals and sound effects in games. Besides that, it was very efficient and easy to use. This research was made with the aim of designing and building an Adventure RPG game using the Perlin Noise algorithm, as well as measuring the level of player satisfaction with the game that has been made. The game engine used in the process of making this game is the 2019 version of Unity Engine 3D with the C# programming language on the windows platform. After the game creation process is complete, it will be continued by testing using the Game User Experience Satisfaction Scales 18 (GUESS-18) method which provides 18 questions to measure the level of player satisfaction with games built with PCG. Based on the results of the tests that have been carried out, the final result with a value of 81,85 % indicates that the game that has been successfully built is included in the good predicate category.

Keywords: *Role Playing Game, Procedural Content Generation, Perlin Noise, Game User Experience Satisfaction Scales*

1. INTRODUCTION

The development of computer technology in the current digital era is very fast. The technology is not only used for information or communication needs, but also for entertainment needs. This has led to the rise of several creative industry sectors, one of which is the game industry in producing video game content [1]. Video games are entertainment in the form of interactive digital that can be played by players using various tools such as PC, game consoles, mobile devices or tablets [2].

Based on data obtained from the Newzoo Global Games Market Report, the number of players playing games in 2020 worldwide was 2.69 billion players with a total revenue of $159.3 billion. In 2021 it will increase to 2.81 billion and it is estimated that in 2023 it will increase again to 3.07 billion players, this is based on an estimated steady increase from 5.6% annually [16]. In video games there are various types of categories such as action, adventure, role playing games, first person shooter and strategy [3].

Role Playing Game (RPG) is one of the most popular game genres. RPG games generally have missions that must be completed by interacting with Non Playable Characters (NPCs), exploring the in-game world, fighting monsters or villains and collecting certain items [3]. In addition, there are key elements of RPG games such as having an interesting story, having character attributes, having quests, having items, experience and levels as well as a combat system [4].

In producing quality game content, it takes a lot of people to be involved with a long production time and requires large storage media, but the demand for quality content continues to increase while game production itself is expensive. In addition, human resources, which is the most important aspect in producing game content are also expensive, slow and the need for human resources will continue to increase over time, resulting in fewer profits from game production [5]. Therefore, a procedural content generation method is needed to create game content using certain algorithms without having to create it manually.

Procedural content generation (PCG) is a method that refers to the automatic creation of game content using a certain algorithm, so that the process of developing game content is not done manually. PCG can create various things in game content such as levels, maps, quests, characters, vegetation, textures [6]. By using PCG, the effort and time needed for game creation will be reduced, the costs needed for game development can be reduced, can lighten the load of memory space on disk, and can provide variety to the development of game content [6].

One of the algorithms that can be applied to map creation is Perlin noise. Perlin noise was discovered by Ken Perlin in 1985 which is used as a smoothing technique for noise which is generally too coarse in the process of making procedural maps [7]. This algorithm is used to create realistic visuals and sound effects in animations and games. Some of the advantages of the Perlin algorithm include: realistic, efficient, easy to use, free of artefacts, applicable in various fields and can be combined with other algorithms. According to Travis Archer, Perlin noise has good quality, requires little memory and is also very fast in making maps [8].

In making a PCG map, there are several algorithms besides Perlin noise such as the diamond square algorithm, midpoint displacement, value noise and simplex noise. But overall using the Perlin noise algorithm will be better in making a map when compared to some of these algorithms because Perlin noise does not require data a long time, does not require large storage, faster and better results [8].

The application of the Perlin noise algorithm has been carried out previously in generating 3D planetary models [9]. In addition, there are other studies in building terrain maps based on data from the Utah part of the United States and making map models resembling hills [10]. Based on this background, the design and development of an adventure role playing game was carried out by applying procedural content generation using the Perlin noise algorithm to facilitate the work in the development and development of game content.

2. METHODOLOGY

A. Game Elements

A game is formed from 2 types of elements that form the basis for making a game concept, these elements are formal elements and dramatic elements. Formal elements are elements that help create the structure of a game. Without these formal elements, a game cannot be called a complete game [2].

Dramatic elements can be defined as elements that involve players to feel the experience of playing the game emotionally. Unlike formal elements, not all dramatic elements have to be included in the game. Dramatic elements themselves can be integrated with formal elements to make the game experience more meaningful for players [11].

B. Procedural Content Generation

In general, PCG in the world of video games is a relatively new field because PCG is mostly used in formal education for computer scientists and for research. PCG has a pyramid-shaped hierarchy which shows that the lower the position, the more algorithms are ready to be used, on the other hand, the higher the position the fewer algorithms that can be used because the system is still experimental and difficult to implement [12].

1) Game Bits are the basic unit of game content. Game bits usually will not involve the user directly, if the game bits can stand alone. Examples of game bits usually include textures, architecture, vegetation, sound, and graphic effects such as fire, water, etc.
2) Game Space is the environment in which the game takes place. An example of a game space is a map or terrain.
3) Game System is a system that simulates a more complex environment. Examples of game systems include ecosystems, roads, and urban environments.
4) Game Scenarios arrange previous levels to be according to plan or according to a sequence of events such as stories, puzzles and levels.
5) Game Design is a rule and mechanism of a game.
6) Derived Content can be created as a companion to the game world, such as leaderboards and items that cover player actions in the game world.

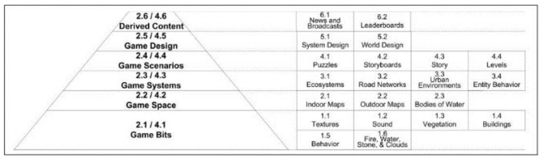


Fig. 1. Classification of game content types on PCG [13].

C. Perlin Noise

Perlin noise is an algorithm to generate noise by sampling and interpolating points in a vector grid at random [14].

The implementation of Perlin noise can be done by determining the set of all points, namely x, y and z where the coordinate points are integer values, then all the points that have been defined are entered into 4 numbers a, b, c, d pseudo random. Perlin noise will produce natural structures such as the creation of mountains [14].

Perlin noise algorithm has an output value that can be represented by color, where a value close to 0.0 is indicated by a darker color, and a value close to 1.0 for a bright color [14].

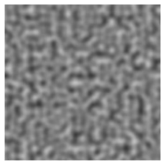


Fig. 2. Perlin noise results [14].

D. Game User Experience Satisfaction Scale – 18

Game User Experience Satisfaction Scales is a validated and comprehensive game scale method with the aim of measuring and evaluating a game. GUESS 18 only has 18 questions based on 9 subscales, namely Usability/Playability, Narratives, Play Engagement, Enjoyment, Creative Freedom, Audio Aesthetics, Personal Gratification, Social Connectivity, and Visual Aesthetics. GUESS-18 has a 7-point Likert scale ranging from strongly disagree with a score of 1 to strongly agree with a score of 7 [15].

The assessment is done by finding the average score of each subscale and then the average score of each subscale can be added up to get the results of the level of player satisfaction with the game [15].

1. Game Design

A. Game Elements

The following are the formal elements and dramatic elements used in the game.

1. Formal Element

a) Player

This game is a game with a single player system, which means it can only be played by one player.

b) Objectives

The main objective of this game is that the player must complete the quest or task given by the NPC.

c) Procedures

- Player run and enter the game, then a splash screen page will appear.
- After that the player enters the start menu page. Here player can select press start to enter the main menu.
- In the main menu, player are given several choices, player can choose play to enter the story scene. Player can also view the help page, credit, or choose exit to exit the game.
- If the player chooses to play, the game will start and the player will be directed to the story scene.
- After the story is complete, players will be directed to the city. Here players can walk around and can receive assignments from NPC.
- After receiving the task, the player can enter the portal where when inside the portal the player can enter the scene map generator which is a world generated using PCG with the Perlin noise algorithm. Every time the player enters the world, the shape of all objects will also change. Here players can defeat existing enemies using the skills that have been provided and players can also complete tasks that have been given by NPCs.
- To complete the game the player must collect the 5 available magic divine items, and if the player manages to collect the five divine magic items, the player will be directed to the scene ending story, ending, ending credit and after that the game will be over. Then the player will return to the main menu.
- If the player's blood has reached 0, then the game will end and the game over page will appear. In this game over panel, players can choose to restart the game or return to the main menu. If the player choose restart then the game will be restarted from the scene map generator and will create a new world. If the player chooses the main menu, the player will return to the main menu.

d) Rules

- To play this game, players must use a mouse and keyboard.
- Player cannot exit when already in the game.

- Player cannot change the shape of the generated world.
- This game only provides 1 task, which means that player can only receive 1 task from an NPC.
- Player can defeat existing enemies by using the skills provided.
- Player and enemies will die when the amount of blood they have reaches 0.
- Players must collect 5 magic divine items to complete the game.

e) Resources

- Player Health: Is an indicator of the blood owned by the player. The player's blood can increase if the player casts skill 5. The player's blood can also decrease when the enemy's attack hits the player..
- Player Skills: Shows 6 skills that can be used by player to defeat all existing enemies. Player can issue these skills by pressing buttons 1 to 6 on the keyboard. The six skills have different cooldowns.
- Quest Box: Shows information in the form of names and indicators of the number of magic divine items that the player must collect. Every time a player manages to get 1 magic divine item, the indicator of the number of magic divine items will also increase.

f) Conflict

The difficulty that occurs in the game is when the player's condition is trying to get all magic divine items to complete the game, but at the same time the enemy is also trying to attack and kill the player.

g) Boundaries

This game has a limitation that this game only provides 1 task that can be accepted by player through NPC. In addition, another limitation is that when the player is already in the gameplay, the player cannot immediately exit the game until the player successfully completes the game or when the player loses. In addition, player can only walk around the area that has been provided in the game.

h) Outcome

Player can be said to have completed the game when they have managed to collect the 5 magic divine items available in this game.

2. *Dramatic Element*

a) Challenge

The challenge in this game is how the player can defeat and survive from enemy and boss attacks without death. Fighting the boss will be more difficult than fighting the enemy because the boss itself has a special attack that can do more damage to player, in contrast to enemies who only have basic attacks. Besides that, another challenge is when player are looking for magic divine items that have been randomly distributed from a fairly wide world.

b) Play

In this game, player are given the freedom to carry out the activities they want to do, such as being free to move or explore anywhere that has been provided in the game, free to carry out missions in a randomly generated world, free to try to fight enemies or not, and player also given the freedom to try all available skills that can be used by player in the game.

c) Premise

This game does not have premise elements.

d) Character

The character in this game is a knight named Black Knight, this knight is equipped with 6 skills that can be used to attack enemies in the game.

e) Story

Tells the life of the terraworld world that is not doing well. It was caused by the attacks of dangerous monsters that made the terraworld world to be destroyed. The monsters not only attack but take something important to the world of terraworld. They had taken all the magic divine which was the thing to maintain the balance of the terraworld world. This makes all residents experience panic and fear. Hearing this a knight came to the terraworld to help the citizens to defeat all the monsters and reclaim all the divine magic that had been stolen.

B. *Flowchart*

The following is a display of some of the flowcharts used to explain the process flow in making game.

1) *Main Menu*

Figure 3 shows a flowchart of the main menu process flow which begins with the display of the splascreen. After that the player will go to the start menu page and the main menu. On the main menu page there are 4 menu options that players can choose from.

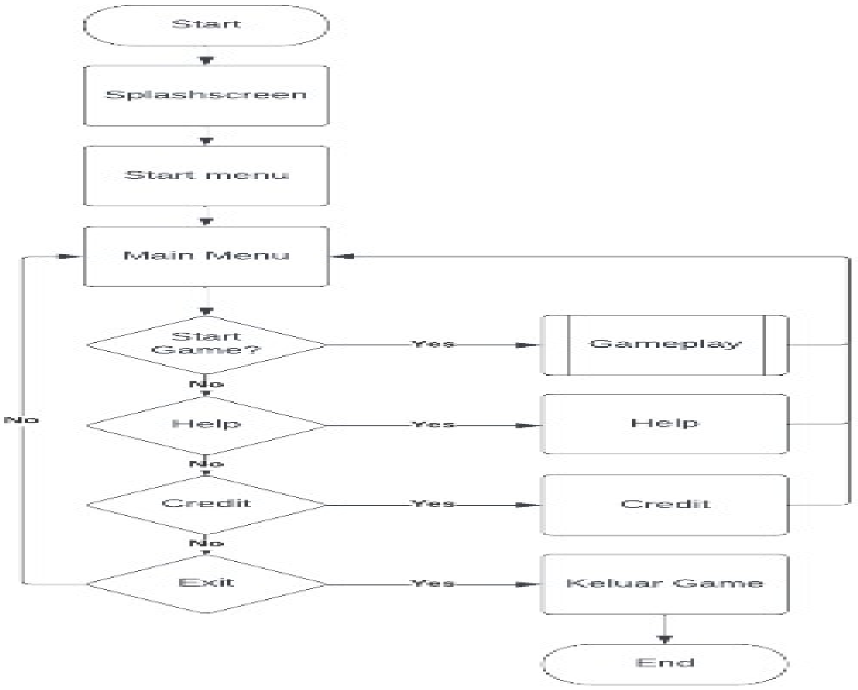


Fig. 3. Main Menu Flowchart

2) *Gameplay*

The gameplay flowchart explains the flow of RPG games with PCG. The process begins with the story and enters the town scene. Here players can take quests from NPCs and enter the portal. After that the player can enter the map generator which is a world created using PCG. If the player loses, a game over display will appear which provides 2 button options, namely restart and main menu. When all items have been collected, the ending story, ending, ending credit pages will appear and will return to the main menu page.

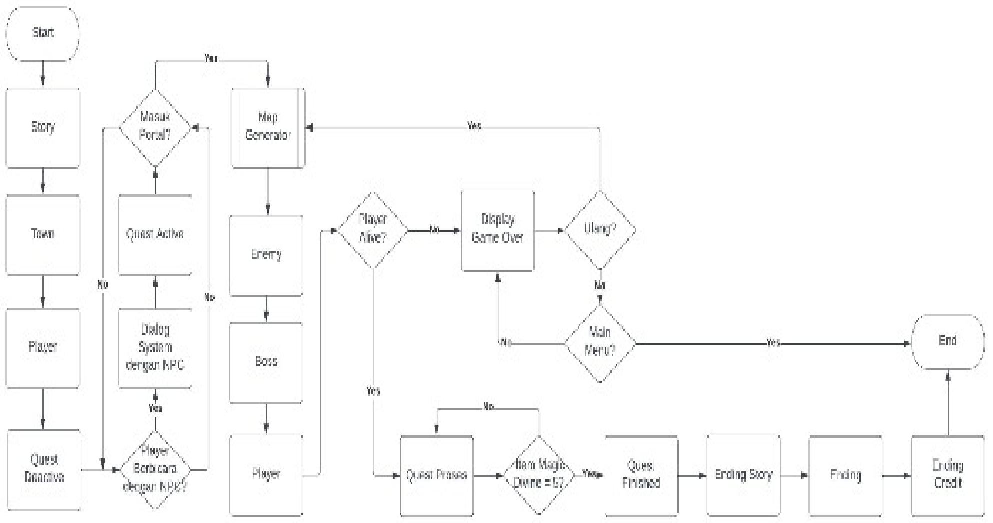


Fig. 4. Gameplay Flowchart

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

3) *Map Generator*

The map generator flowchart starts the process by creating an enum draw mode to display the map results on the screen. Then do the initialization of variables on the map generator taken from other scripts such as map width, map height and others. After that, it is continued by doing iterations in the region where the function of this region is the stage of giving color to each pixel from the noise results which are represented by a value range of 0 to 1 based on the color of the region that has been determined. The color assignment process is carried out using the color function and at this stage the map width and map height calculations are also carried out to determine the number of arrays needed to apply regional colors to all map dimensions. In the map generator function, you can select the map results you want to display, such as a noise map, color map and mesh. In addition, there is a validation process for map height, map height and octaves to avoid inputting errors on the map that has been created. After that, a random system process is carried out on the seed object based on the value range from 1 to 200 so that later there will be 200 different map shapes that can be displayed on the screen. The generate map process is also called into the start function so that every time a player starts the game, the map that will be displayed will change according to the seed value.

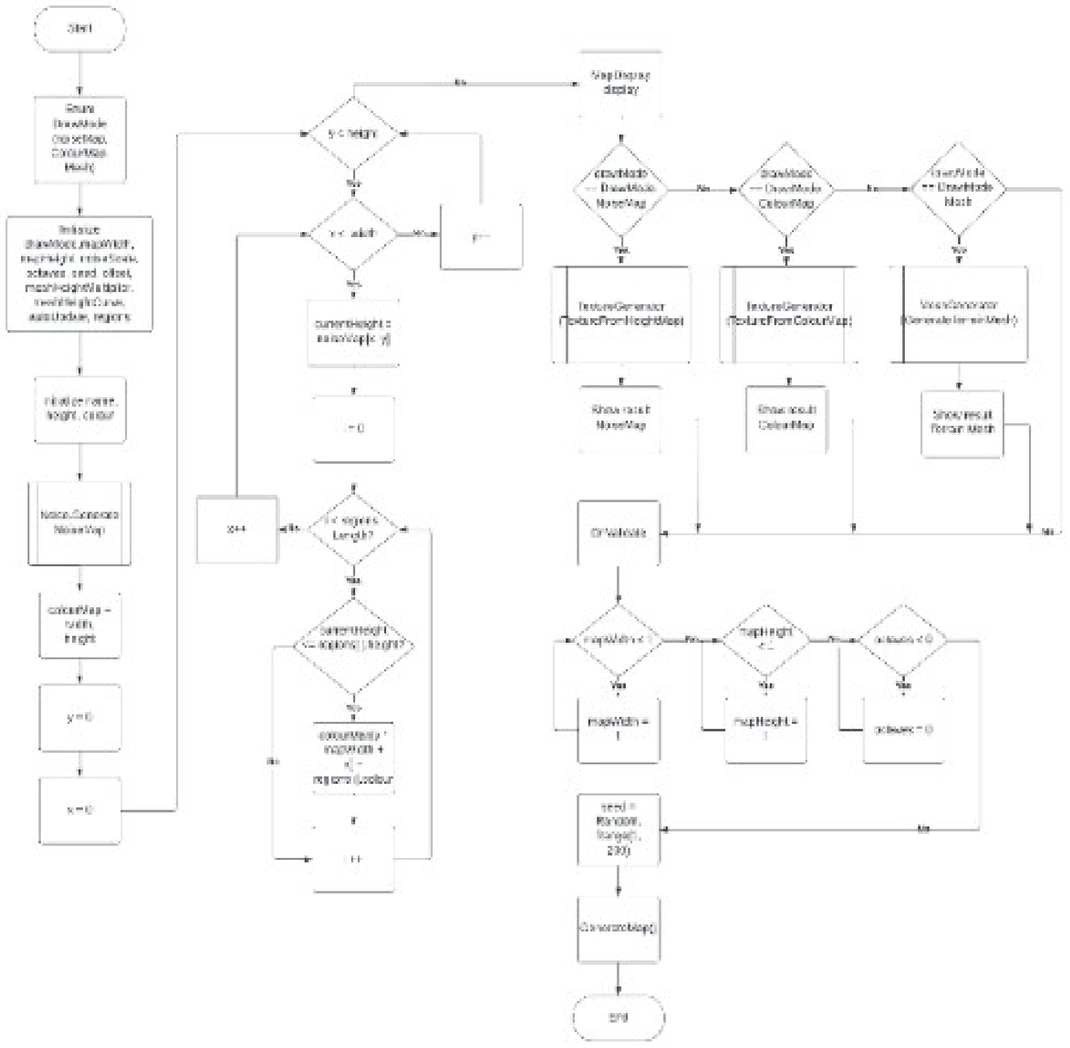

Start
Enum DrawMode (NoiseMap, ColourMap, Mesh)
Initialize drawMode, mapWidth, mapHeight, noiseScale, octaves, seed, offset, meshHeightMultiplier, meshHeightCurve, autoUpdate, regions
Initialize name, height, colour
Noise.GenerateNoiseMap
colourMap = width, height
y = 0
x = 0
y < height
x < width
y++
currentHeight = noiseMap[x, y]
i = 0
i < regions.Length?
x++
currentHeight <= regions[i].height?
colourMap[y * mapWidth + x] = regions[i].colour
i++
MapDisplay display
drawMode == DrawMode.NoiseMap
drawMode == DrawMode.ColourMap
drawMode == DrawMode.Mesh
TextureGenerator (TextureFromHeightMap)
TextureGenerator (TextureFromColourMap)
MeshGenerator (GenerateTerrainMesh)
Show result NoiseMap
Show result ColourMap
Show result Terrain Mesh
OnValidate
mapWidth < 1
mapHeight < 1
octaves < 0
mapWidth = 1
mapHeight = 1
octaves = 0
seed = Random.Range(1, 200)
GenerateMap()
End

Fig. 5. Map Generator Flowchart

4) *Noise*

The noise flowchart explains the workings and flow of the Perlin noise algorithm which is the initial basis for generating maps. Then do some initialization of the variables needed to create a map such as map width, map height, seed, scale, octaves and offset. Map width and map height are used to determine the size of the map, seed as an object generated to generate a map, scale to set the size of each pixel resulting from Perlin noise that you want to display on the screen, octaves to adjust the level of detail from Perlin noise results and offset to get the same map shape with different positions.

After that, the validation process is carried out on the scale variable to overcome the division error with the number 0. There are half width and half height functions used so that when scaling the map results it is done from the middle position.

After that, looping on the variables height, width and octaves is carried out to determine the size of the map and after that it is continued by performing calculations using the Matfh.PerlinNoise function from the unity engine library. The calculation results will be stored in the map results at all coordinate points.

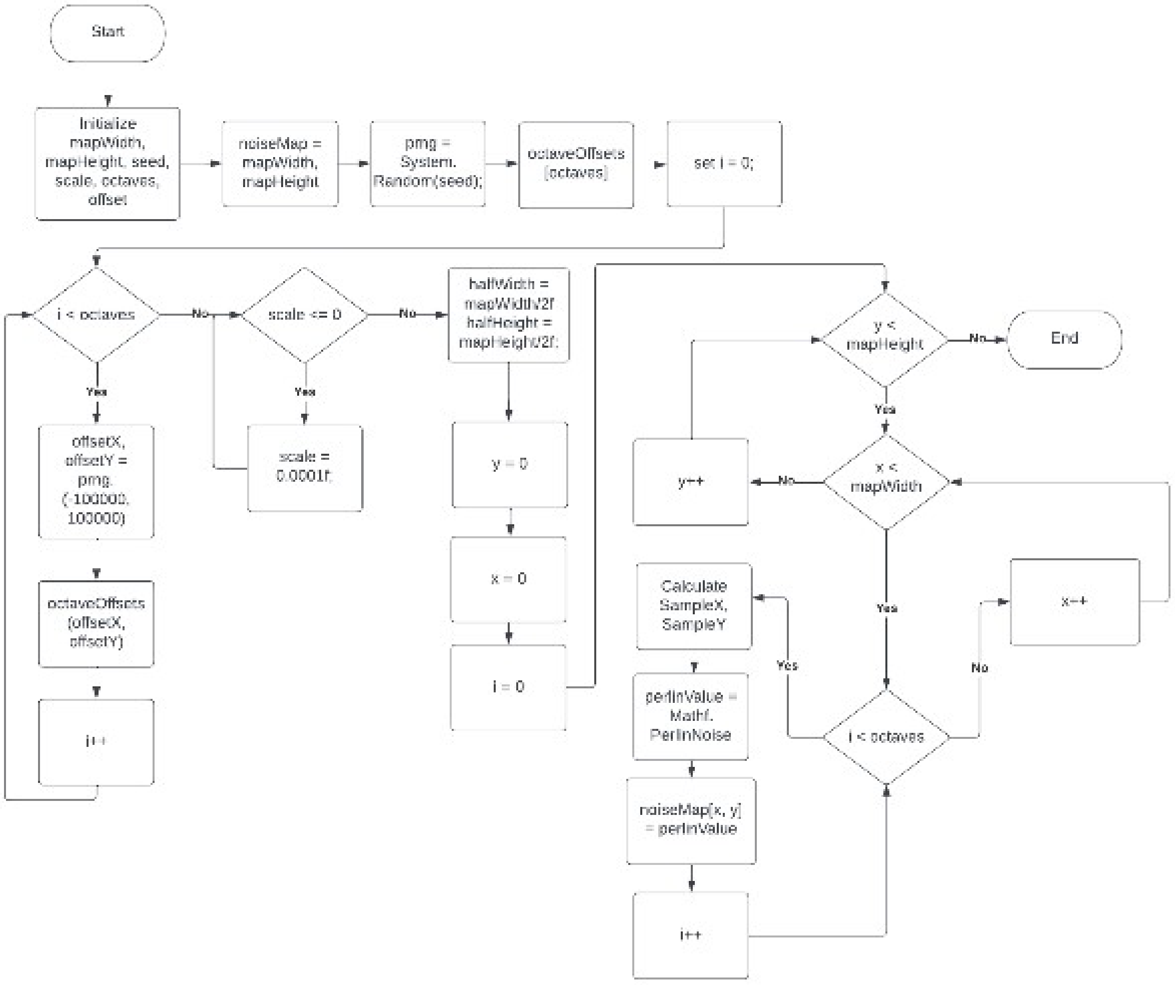

Start
Initialize mapWidth, mapHeight, seed, scale, octaves, offset
noiseMap = mapWidth, mapHeight
prng = System. Random(seed);
octaveOffsets [octaves]
set i = 0;
i < octaves
No
Yes
scale <= 0
No
Yes
halfWidth = mapWidth/2f halfHeight = mapHeight/2f;
offsetX, offsetY = prng. (-100000, 100000)
scale = 0.0001f;
y = 0
x = 0
i = 0
octaveOffsets (offsetX, offsetY)
i++
y < mapHeight
No
End
Yes
y++
No
x < mapWidth
Yes
Calculate SampleX, SampleY
perlinValue = Mathf. PerlinNoise
noiseMap[x, y] = perlinValue
i++
Yes
i < octaves
No
x++

Fig. 6. Noise Flowchart

5) *Map Display*

The map display flowchart shows the process of displaying the results of Perlin noise into the board object in the game environment which is stored by the texture renderer variable. After that, the process of adjusting the size of the board object is carried out so that its size is the same as

the texture containing the Perlin noise by using the local scale transform function.
In addition, there is a mesh filter that is used to retrieve the existing mesh on the board object containing the terrain map results, mesh renderer to display the terrain map results to the screen and a collider mesh that is used so that all objects in the game can come into contact with the terrain map results.

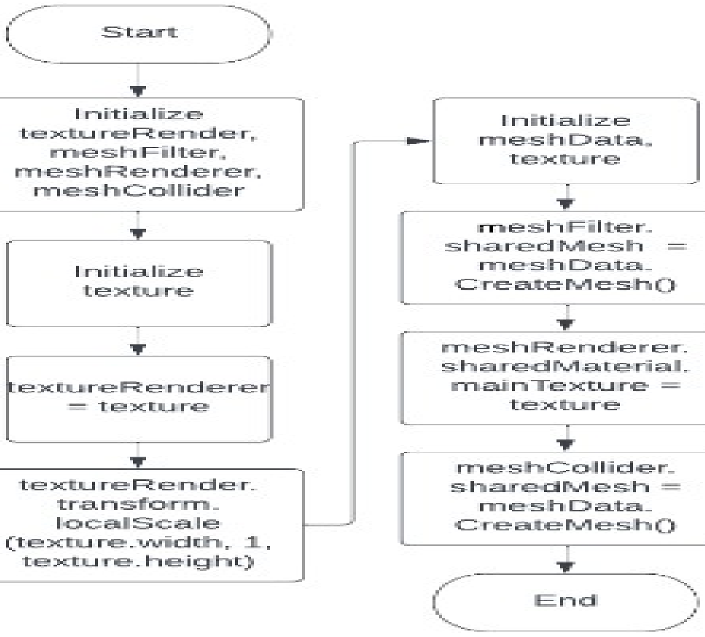

Start
Initialize textureRender, meshFilter, meshRenderer, meshCollider
Initialize texture
textureRenderer = texture
textureRender. transform. localScale (texture.width, 1, texture.height)
Initialize meshData, texture
meshFilter. sharedMesh = meshData. CreateMesh()
meshRenderer. sharedMaterial. mainTexture = texture
meshCollider. sharedMesh = meshData. CreateMesh()
End

Fig. 7. Map Display Flowchart

6) *Texture Generator*
The texture generator flowchart explains the process of giving textures in black and white to the resulting Perlin noise. In this process, there are filter mode and wrap mode functions that are used to fix the level of blur in the Perlin noise texture. The color assignment process is carried out with the color function on the color map variable and the color lerp function so that the resulting color becomes smoother.

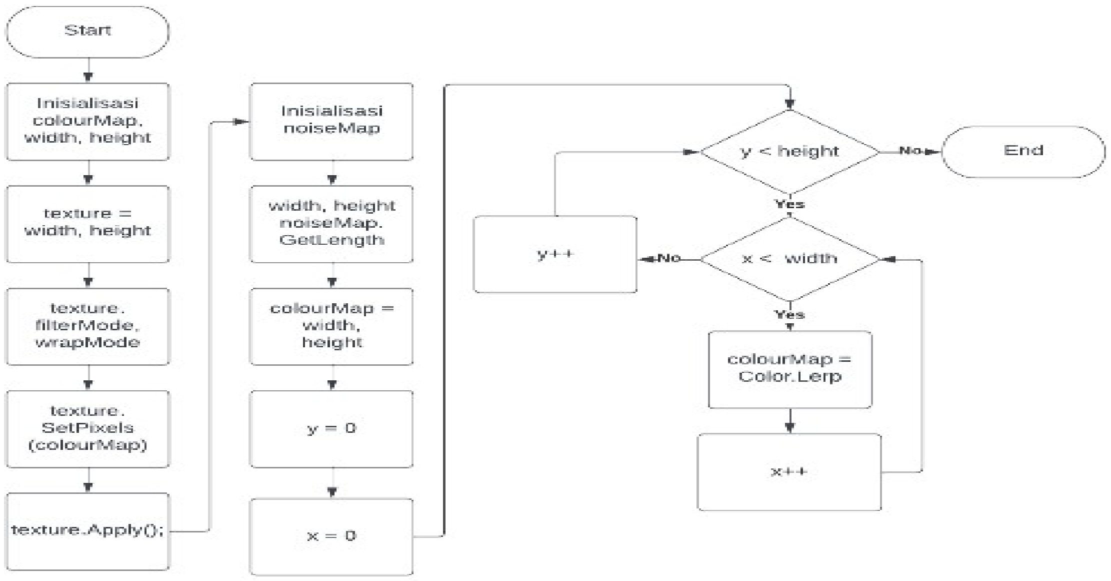

Start
Inisialisasi colourMap, width, height
texture = width, height
texture. filterMode, wrapMode
texture. SetPixels (colourMap)
texture.Apply();
Inisialisasi noiseMap
width, height noiseMap. GetLength
colourMap = width, height
y = 0
x = 0
y < height
No
End
Yes
x < width
No
y++
Yes
colourMap = Color.Lerp
x++

Fig. 8. Texture Generator Flowchart

7) *Mesh Generator*

The mesh generator flowchart explains the process of creating a terrain map using a mesh system. It starts with initializing the vertices variable which is the points to create a mesh, triangle which is used to form a triangle from a collection of vertices, uvs which regulates the texture and triangle index which is the storage medium for each vertices.

The process of adding triangle method will be executed by taking 3 integers. After that, it will go to the next process, namely triangles[triangleIndex] = a, which means that the value of a is the first vertices added and stored into the triangle array index to 0. After finishing creating the first vertice, it will continue with the next process, namely triangles[triangleIndex. + 1] = b, which means that the value of b is the second vertice that is added and stored into the 1st index triangle array. After finishing creating the second vertice, it will continue with the next process, namely triangles[triangleIndex + 2] = c, which means the value c is the third vertice that is added and stored into the index triangle array 2. Then after finishing creating the three vertices that require these three arrays, it is necessary to add 3 more empty arrays so that other triangles can be filled with 3 vertices as well.

In the mesh generator there is also a height multiplier and height curve variable that is used to adjust the height of the terrain map so that the shape is not just flat. After completing the add triangle system to form a triangle, then proceed with the process of merging the first triangle with the second triangle to form a square

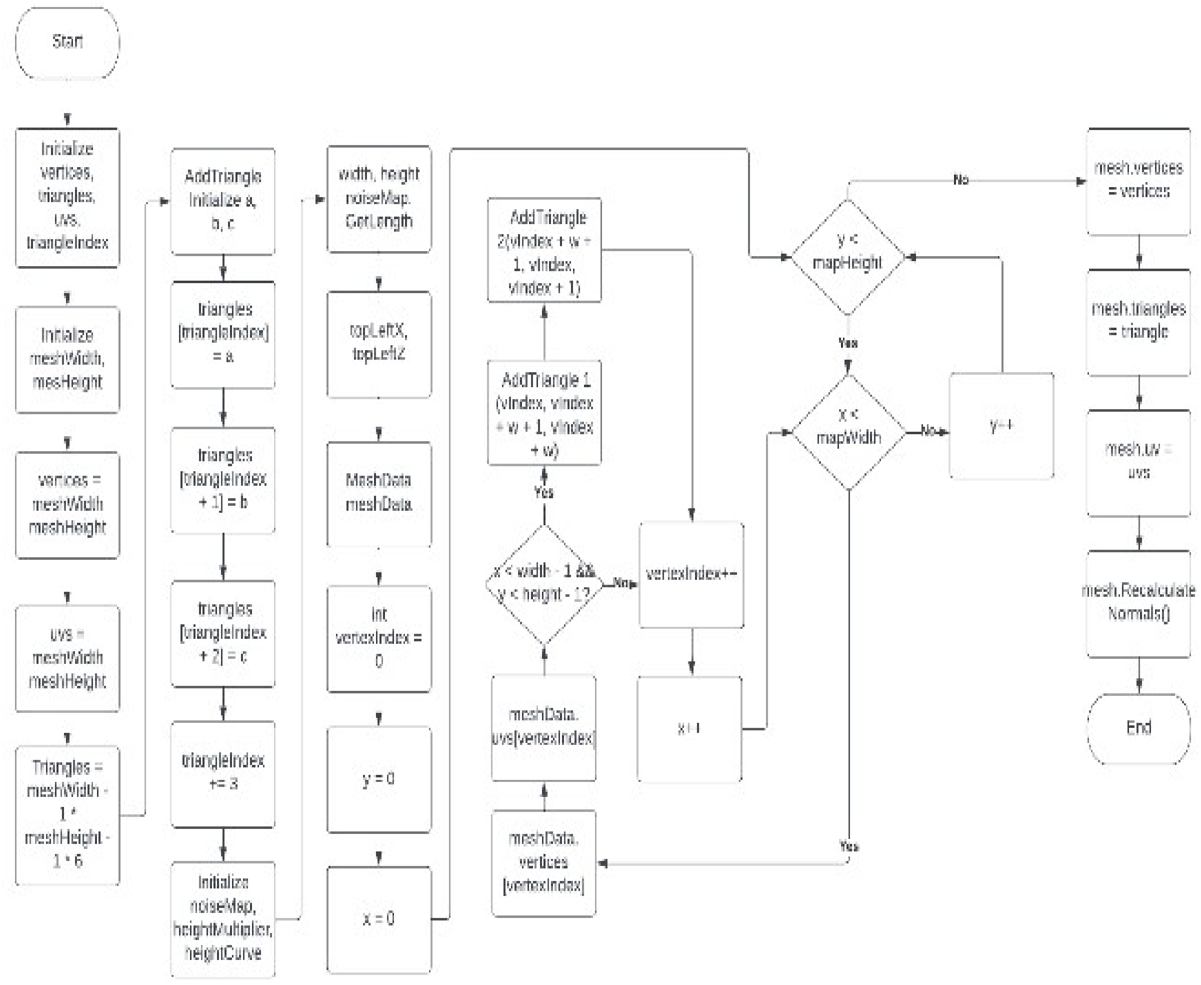

Start
Initialize vertices, triangles, uvs, triangleIndex
Initialize meshWidth, mesHeight
vertices = meshWidth meshHeight
uvs = meshWidth meshHeight
Triangles = meshWidth - 1 * meshHeight - 1 * 6
AddTriangle Initialize a, b, c
triangles [triangleIndex] = a
triangles [triangleIndex + 1] = b
triangles [triangleIndex + 2] = c
triangleIndex += 3
Initialize noiseMap, heightMultiplier, heightCurve
width, height noiseMap. GetLength
topLeftX, topLeftZ
MeshData meshData
int vertexIndex = 0
y = 0
x = 0
AddTriangle 2(vIndex + w + 1, vIndex, vIndex + 1)
AddTriangle 1 (vIndex, vIndex + w + 1, vIndex + w)
Yes
x < width - 1 && y < height - 1?
No
vertexIndex++
x++
meshData. uvs[vertexIndex]
meshData. vertices [vertexIndex]
y < mapHeight
No
Yes
x < mapWidth
No
y++
Yes
mesh.vertices = vertices
mesh.triangles = triangle
mesh.uv = uvs
mesh.RecalculateNormals()
End

Fig. 9. Mesh Generator Flowchart

3. RESULT AND DISCUSSION

A. Implementation

Figure 10 is a display of the splashscreen page. The display of this page contains the game logo and the unity logo. After the splashscreen process has been completed, the player will be directed to the start menu page.

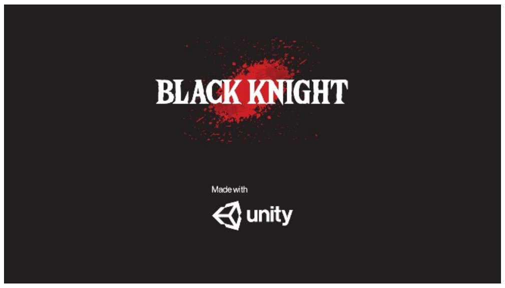


Fig. 10. Splashscreen

Figure 11 is a display of the start menu page. On this page there is a press start button that players can sign. If the player presses this button, they will be directed to the main menu page.

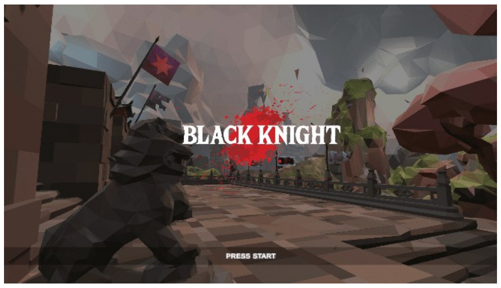


Fig. 11. Start Menu

Figure 12 is a display of the main menu. On the menu there are 4 options, namely play, credit, help and exit. If the play option is selected, the game will start. If the credit option is selected, the player will enter the credit page. If the help option is selected, the player will enter the help page. If the exit option is selected, the player will exit the game.

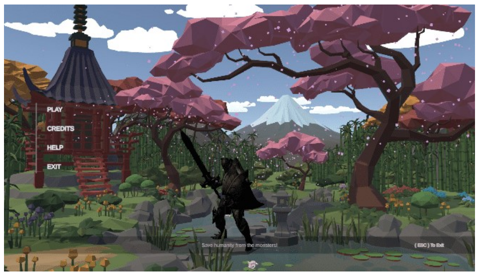


Fig. 12. Main Menu

Figure 13 is a display of the credit page. On this page will display the name of the creator of the game black knight.

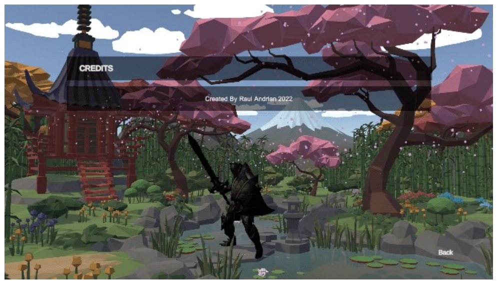
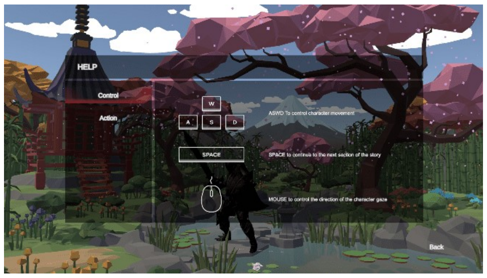


Fig. 13. Credit

Figure 14 shows information about some of the controls that can be used by players when playing games along with an explanation of the use of each of these controls.



ASWD To control character movement
SPACE to continue to the next section of the story
MOUSE to control the direction of the character gaze

Fig. 14. Help

Figure 15 is a view of the story. On this page there is a box containing stories that can be read by players. Players can continue the story until the end by pressing the next button that has been provided.

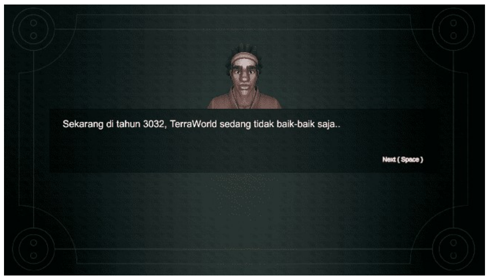

Sekarang di tahun 3032, TerraWorld sedang tidak baik-baik saja..

Fig. 15. Story

Figure 16 is a display of the NPC dialog. When the player approaches the NPC to interact, a dialog box will appear containing the conversation between the NPC and the character. Players can also press the next button that has been provided to continue the contents of the conversation until the end section.

Journal of Theoretical and Applied Information Technology
15th May 2023. Vol.101. No 9
© 2023 Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

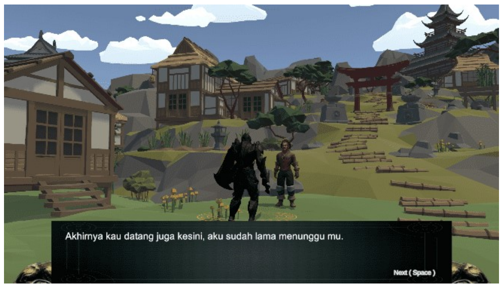

Akhirnya kau datang juga kesini, aku sudah lama menunggu mu.
Next (Space)

Fig. 16. Dialog NPC

Figure 17 shows information related to tasks that must be completed by players such as names and the number of items that must be collected. This image also shows the portal used to enter the world generated with PCG.

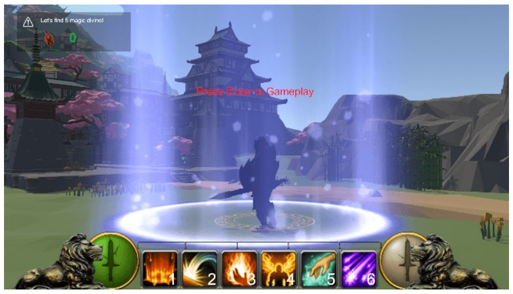


Fig. 17. Quest Active and Portal

Figure 18 is a display of the map results that have been generated with PCG. Coloring is done based on the division of the region that will be in the generation. In this game, the region is divided into 7 sections with a variety of predetermined colors.

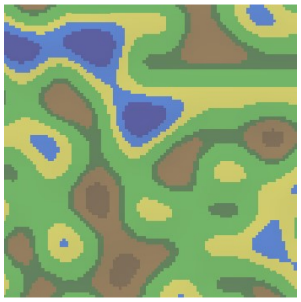


Fig. 18. Map Implementation

Figure 19 is the display when the player attacks the enemy using skill 6. To issue this skill the player can press the 6 button on the keyboard. When hit by an attack from skill 6, the enemy's blood will decrease. Skill 6 has greater damage and a longer cooldown system compared to the previous skill.

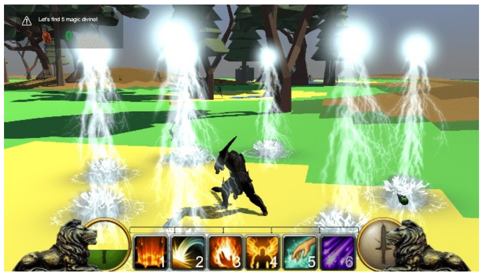


Fig. 19. Skill 6 Ultimate

Figure 20 is the display when the player finds a divine magic item. To retrieve items the player can press the F key on the keyboard. After the item is successfully retrieved, the number of items collected will increase.

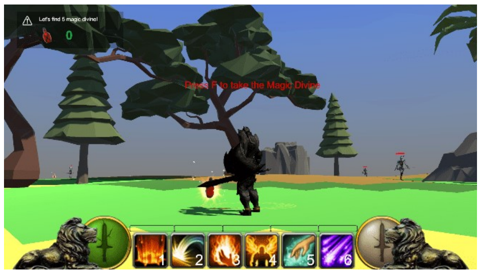


Fig. 20. Item Magic Divine

Figure 21 is a display of the game over page. This page will be displayed when the player has been defeated by the enemy. There are 2 options, namely restarting or main menu. Restart to repeat the game and main menu to return to the main menu page.

Fig. 21. Game Over

Figure 22 is a display of the ending credits page. The ending credits page will display all the names of the contributors who played a role and contributed to the making of the game, including thanks for everyone who has played this game. After the ending credits process is complete, players will be directed back to the main menu page.

Terima Kasih Sudah Bermain

PENGEMBANG

Programmer	Raul Andrian
Designer	Raul Andrian
Project Manager	Raul Andrian

Didukung Oleh

Fig. 22. Ending Credits

B. *User Satisfaction Test*

After the game is successfully built, a testing process is carried out where the respondents will play the games that have been designed and built. After finishing playing the game, the respondents will be asked to fill out a questionnaire that has been created using GUESS-18. The results of the questionnaires that have been filled out by the respondents will be used to measure the level of player satisfaction with the games that have been built.

This study managed to collect 36 respondents who have played the game and filled out a google form questionnaire.

Table 1. Questionnaire Results

Subscales	**Result**
Usability/Playability	85.3 % (Very Good)
Narratives	86.5 % (Very Good)
Play Engrossment	71.2 % (Pretty Good)
Enjoyment	79.45 % (Good)
Creative Freedom	78.5 % (Good)
Audio Aesthetics	81.9 % (Good)
Personal Gratification	84.3 % (Very Good)
Visual Aesthetics	87.65 % (Very Good)
Final Result	81.85 % (Good)

Based on the results of calculations that have been carried out on the questionnaire that has been filled out by the respondents, the final result is obtained with a value of 81.85%, which means that in testing the level of player satisfaction with RPG adventure games built with PCG using Perlin noise, it falls into the category of good predicate.

Although it produced good quality of RPG adventure games, it had some complexities of algorithm, including time and memory requirement, moreover significant computation. The difference from prior work, it was not only enhanced display of realism as in [16], but also other scales as in Table 1. The scientific contribution of this research are realistic visuals and sound effects with good predicate of player satisfaction.

4. CONCLUSIONS AND SUGGESTIONS

A. *Conclusions*

Based on the research that has been done, it is concluded that the Perlin Noise algorithm has been successfully applied in the design and development of an Adventure RPG game using the 2019 version of Unity Engine 3D and the C# programming language on the windows platform. This game that has been successfully built implements several features such as player skills, items, quest systems, battle systems and various other features that are made to give the impression of an interesting and fun playing experience for players when playing this game. The implementation of the Perlin Noise algorithm is used to generate a map in the form of terrain whose shape will always change automatically, so that it can provide a new experience and atmosphere every time the player starts the game.

After the game design and development process has been completed, a build project for the windows platform is carried out and tested using the GUESS-18 method. Based on the results of the tests that have been carried out, a total of 36 samples of respondents were obtained with a value of 81.85% which indicates that players are satisfied with the game that has been built using the PCG method using the Perlin Noise algorithm. These results are obtained from the calculation of the average based on the value of each subscale that has been answered by the respondents. The results with a figure of 81.85% indicate that the game that has been built is included in the good predicate category.

B. Suggestions

Based on the research that has been done, some suggestions that can be used as input for further research:

1. In this study only focuses on the Perlin noise algorithm, therefore it is hoped that further researchers can develop by implementing the marching squares algorithm in the process of forming the surface on the terrain map where this is done so that the appearance of the terrain map shape generated by the Perlin noise algorithm does not looks too stiff and becomes smoother.
2. Adding variations in the game space or environment where the game is in the game that can be explored by players, both in the city and in the map area created by PCG so that it can improve the player's experience when playing the game.

3. In making maps using PCG, it is recommended to add more variations of game objects such as houses, vegetation, weapons, adding textures and other game objects. This is done in order to avoid the appearance of game content that feels too similar to one another every time a player enters the game, even though the game objects have the same function or role with a difference in the visual form of the game content, which is expected to create a visual appearance of the game becomes more interesting and does not make the players feel bored while playing the game.

ACKNOWLEDGMENTS

Acknowledgments This research can be carried out properly thanks to the assistance of Multimedia Nusantara University. Thank you for the support and assistance that has been given during the process of writing this article.

REFERENCES:

[1] C. Fajri, “Tantangan Industri Kreatif-Game Online di Indonesia,” *J. ASPIKOM*, vol. 1, no. 5, p. 443, 2012, doi: 10.24329/aspikom.v1i5.47.
[2] W. Istiono, Hijrah, and N. N. P, “Education Games To Learn Basic Algorithm With Near Isometric Projection Method,” *Int. J. Adv. Stud. Comput. Sci. Eng. IJASCSE*, vol. 8, no. 7, p. 5, 2019, [Online]. Available: http://arxiv.org/abs/2005.13225%0Ahttp://dx.doi.org/10.31227/osf.io/yuzn7
[3] S. Björk and J. P. Zagal, *Game Design and Role-Playing Games*, no. April 2018. 2018. doi: 10.4324/9781315637532-18.
[4] E. Adams and J. Dormans, *Designing Game Mechanics*. 2012.
[5] G. Smith, *Procedural Content Generation*. 2016. doi: 10.1201/9781315313412-9.
[6] J. Togelius *et al.*, “Procedural Content Generation : Goals, Challenges and Actionable Steps,” *Artif. Comput. Intell. Games*, vol. 6, no. July, pp. 61–75, 2013, [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2013/4351%5Cnhttp://drops.dagstuhl.de/opus/volltexte/2013/4336/
[7] I. Parberry, “Designer Worlds: Procedural Generation of Infinite Terrain from Real-World Elevation Data,” *J. Comput. Graph. Tech.*, vol. 3, no. 1, pp. 74–85, 2014, [Online]. Available: http://jcgt.org/published/0003/01/04/
[8] T. Archer, “Procedurally Generating Terrain,” *44th Annu. midwest Instr. Comput. Symp. Duluth*, pp. 378–393, 2011.
[9] Vuontisjärvi H, “Procedural Planet Generation in Game Development,” 2014.
[10] M. Andersson and S. Kvarnström, “Procedural Generation of a 3D Terrain Model Based on a Predefined Road Mesh,” 2017, [Online]. Available: https://gupea.ub.gu.se/handle/2077/53338
[11] S. Putra and W. Istiono, “Implementation Simple Additive Weighting in Procedural Content Generation Strategy Game,” vol. 4, no. 12, pp. 9–18, 2022.
[12] N. A. Barriga, “A Short Introduction to Procedural Content Generation Algorithms for Videogames,” *Int. J. Artif. Intell. Tools*, vol. 28, no. 2, 2019, doi: 10.1142/S0218213019300011.
[13] M. Hendrikx, S. Meijer, J. Van Der Velden, and A. Iosup, “Procedural content generation for games: A survey,” *ACM Trans. Multimed. Comput. Commun. Appl.*, vol. 9, no. 1, 2013, doi: 10.1145/2422956.2422957.
[14] K. Perlin, “Image Synthesizer.,” *Comput. Graph.*, vol. 19, no. 3, pp. 287–296, 1985, doi: 10.1145/325165.325247.
[15] J. R. Keebler Assoc, W. J. Shelstad, D. C. S. Google, B. S. Chaparro, and M. H. Phan Google, “Validation of the GUESS-18: A Short Version of the Game User Experience Satisfaction Scale (GUESS),” *J. Usability Stud.*, vol. 16, no. 1, pp. 49–62, 2020.
[16] A. K. Ginting, K. Sari, C. Fadhilah, R. N. Yusra, D. Hartama, and M. Zarlis, “Application of the Perlin Noise Algorithm as a Track Generator in the Endless Runner Genre Game,” *J. Phys. Conf. Ser.*, vol. 1255, no. 1, 2019, doi: 10.1088/1742-6596/1255/1/012064.