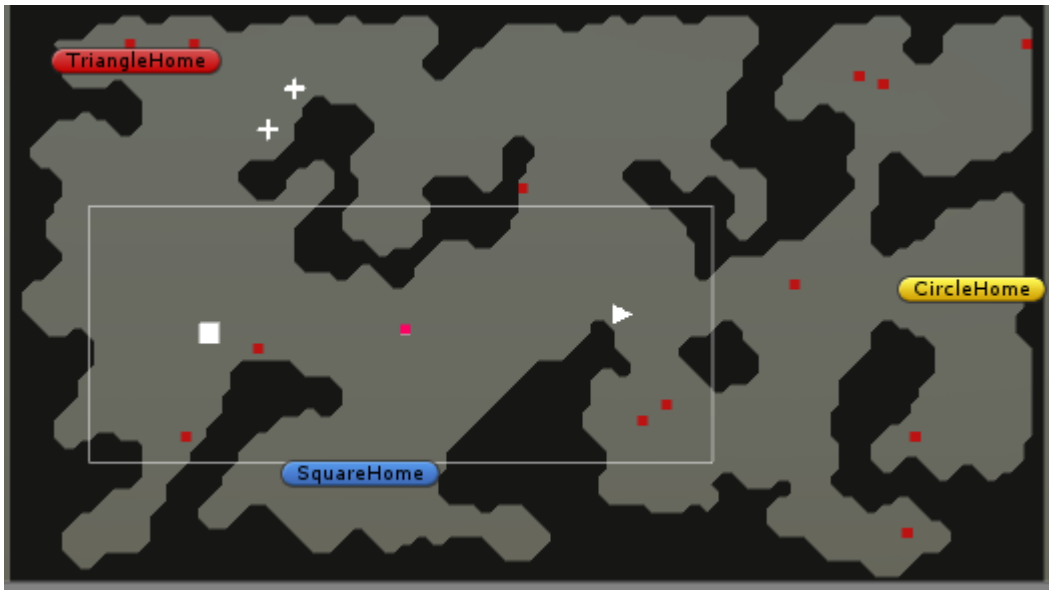
COM569 ASSIGMENT 3

DESIGN REPORT

VASILEIOS DRETTAS



Contents

Introduction

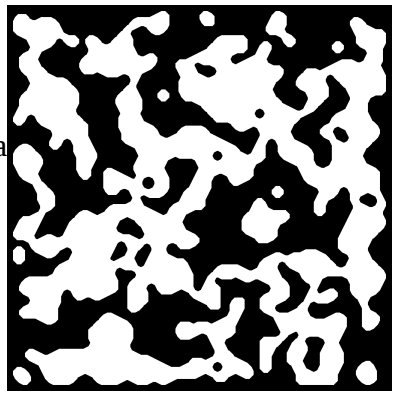
This report focuses on how to build a simulation gaming world with simple geometric shapes as the basis for characters and other game world elements. Geometric shapes have different phycsical properties and as a result can have different functions and attributes in our game. The gaming environment is procedurally generated bases on the cellular automata model and have shapes like circles, squares, crosses and triangles which are interacting with each other. Also the shapes are interacting with the environment and have functions like hunt for food, go home, and explore the world.

Influences and references.

Geometric shapes

In order to build a game with simple geometric shapes I studied some papers and researches about the human emotions for those shapes. For example researches have showed that downward-pointing triangles are associated with threats and are more intimidating than upwards pointing triangles. This kind of information can be found in there [1] and [2]. Games like “Geometry dash” and “Geometry wars” use simple geometry shapes for the environment and the characters.

Procedurally generated map



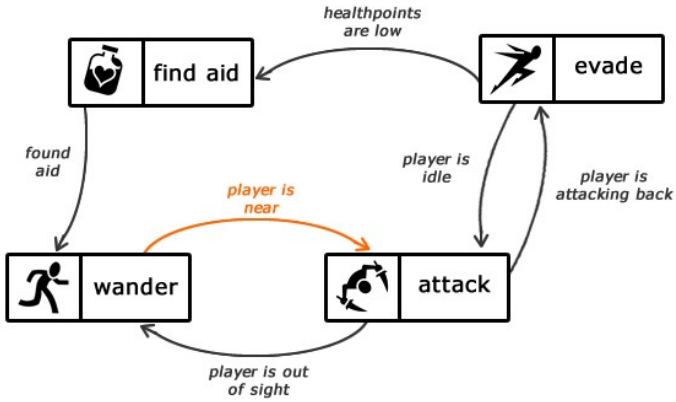
In order to build a procedurally generated map i used the cellular automata model. With cellular automata cave generation, my game map is every time unique, so the player will every time immerse a completely different experience. Useful information about cellular automata can be found in [3] and [4]. Also Sebastian Lague youtube channel has some tutorials on how to generate a cellular automata cave[5]. Games like “Minecraft” and “Spelunky” have procedurally generated maps.

Pathfinding

For the movement of the shapes in the world I had to implement a pathfinding algorithm. With the pathfinding algorithm the shapes will go to their destination from the closest path and they will avoid any obstacles. There are a lot of pathfinding algorithms like dijkstra, prim, kruskal shortest path algorith[6][7][8]. Instead of those three i choose the A* pathfinding algorithm because it is the most efficient. Useful information about how to implement A* pathfinding in a unity game can be found in tutorials [9].

Finite state machine AI

In our game world we want our shapes to make decisions according to their situation and interact with the other shapes of the world. To achieve that I had to implement some kind of Artificial intelligence to the shapes. One easy way is with FSM (finite state machine). A finite state machine is a device or a model of a device which has a finite numbers of states that it can be in at any given time. A finite state machine can only be in one state at a time. Some useful information about FSM AI can be found here [10] and in this book [11].

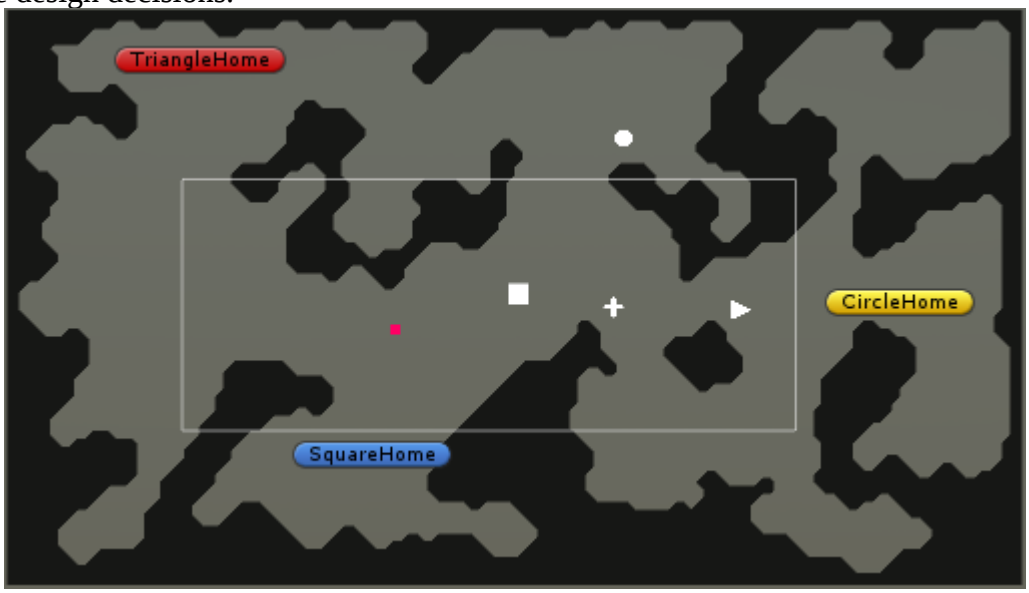


Simulation

Our game world is a simulation. We want the shapes to interact with each other and do some basic functions. Also we want to guarantee some balance in the world, none of the shapes will overpopulate and at the same time none of the shape will extinct. Some famous world simulation games are “The Sims” and “Second life”.

Description of my game

In this chapter I will describe all the properties of the game, the shapes interaction, FSM AI, and i will justify some design decisions.



Starting with some general information I used week 8 lab demo as the basis for my game. In this way I had already a functional procedurally generated map. Also I had already implemented the pathfinding algorithm and some functions of the shapes. Each shape had a far and a near circle collider. Those two sensors are used to detect other shapes and other game world elements.

Shapes

Based on the research papers about the human emotions for shapes, I implement different functions and chose different attribute values in each type of shape. Circles are avoiding conflicts and run fast. Triangles are not as fast as circles but they are aggressive and they try to fight when they have the chance. Squares are slow but very strong. They like to fight too. Lastly crosses have the ability to heal other objects.

Home

Circles, triangles and squares have their own homes. The homes are placed on the boarders of the map because we don't want to have the triangle home close to circles home and make the triangles chase the circles all the time. The homes are also use to give some health to a wounded shape.

Food

Every shape has an attribute named stamina. In order to maintain the stamina the shape has to eat food. In the case that the stamina level drops to zero the shape dies. The food is generated at random points of the map and can be used to maintain the population of the world. This's because we create only 30 food objects on the map. If the shapes overpopulate they will not have enough food and as a result they will die.

FSM

The FSM AI was one of the most difficult parts of my game implementation because each shape interacts differently when it detects another shape. However all the shapes have some general states. We can classify the states in 3 categories nonContact, alertStates, inContact.

NonContact states

Idle: In this state the shape decides with probability 10% to stay at the same state and with probability 90% it will go to the state explore.
Explore: In this state we choose a random point on the map which is ground and set it as a waypoint. Then we set it as a target on the pathfinding script and the shapes starts to moving towards it. When the shape is very close to the waypoint (we calculate the distance between the shape and the waypoint with the Vector3 method Vector3.Distance) we choose a new random point on the map and set that as a waypoint. In this way our shape is exploring the map and hopefully at some point it will detect another shape.
GoHome: In this state the shape set as a target in the pathfinding script its own home. Then the pathfinding script starts moving the shape to the home.
Alert: This state is usually chosen by the shapes when they detect another shape in their far sensor. It doesn't have any functionality but it is required the nonContact state to be alert if we want to choose any alertState or inContact state.

AlertStates (requires the nonContact state to be Alert)

Evade: In this state the shape tries to get away from another shape. It chooses a random point in the map and then it change to state explore to move towards that point.

Chase: In this state the shape has detect another shape and follows it. In order to do that it sets at the pathfinding script the target to the detected object.
Contact: This state is usually chosen by the shapes when they detect another shape in their near sensor. It doesn't have any functionality but it is required the Alert state to be contact if we want to choose any inContact state.

inContact (requires the AlertStates state to be Contact and the nonContact state to be Alert)

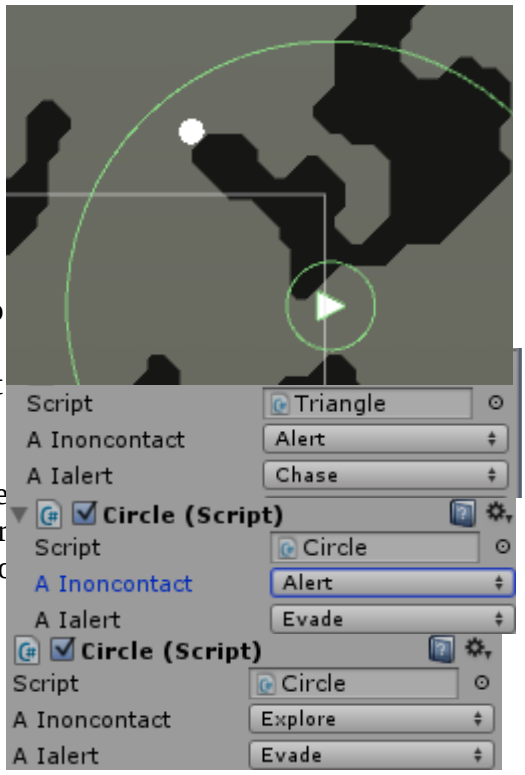
Fight: In this state the shape is fighting the shape which is in its near sensor. It sends a message to the other shape to lose some health (vitality). There is a probability 10% that the shape will do a critical hit and as a result the other shape will lose more health than usual.
Recreate: In this state the shape is creating other shape of the same type.
HealOther: In this state the shape is healing the shape which is in its near sensor. Only the cross has this function.
GetHeal: In this state the shape is getting vitality by a cross.

Decision making

A very important part on the FSM AI is when to change from one state to another. In the game I used mostly the shape colliders to achieve that.

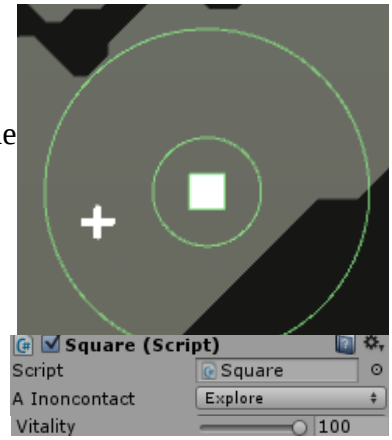
Far sensor

The far sensor (biggest collider) is used to detect shapes and game world elements. As we can see from the picture the triangle saw a circle in its far sensor. At this moment it stops any other function that is currently doing and it is getting to the alert state. Then it has to chose which alert state has to be setted up. A triangle likes to fight. So I decided to generate a random number from 1 to 10 and if the aggression value of the triangle is higher than the generated number it is chasing the shape to fight it. Else it is avoiding the other shape.



From the other hand the circle also detects in its far sensor the triangle alert. After it sees that the detected object is a triangle and sets the aler doesn't like conflicts. The evade state as we said before changes the no

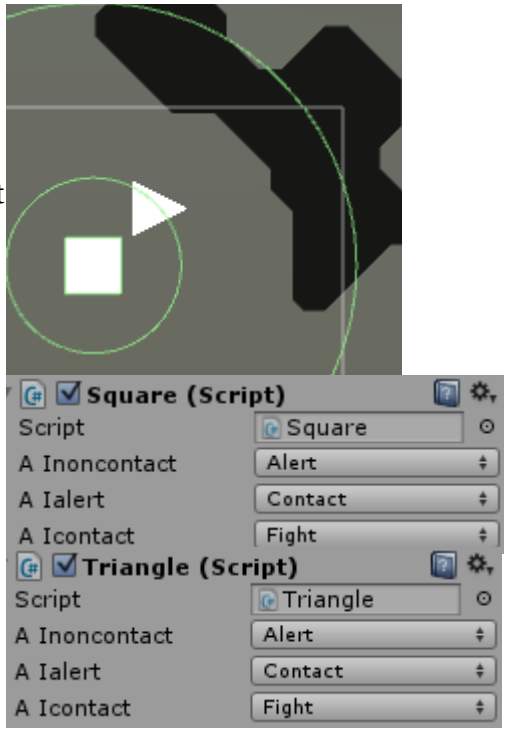
The far sensor is more or less the same to the square and the triangle. When they detect a cross and they need health, they are chasing it to increase their health. When they detect same type shapes and want to reproduce they chase the detected shape. Finally when they detect a circle or a square(in the case of triangle)/triangle(in the case of square) they randomly chose based on their aggression value as i explained before. In the picture we can see that the square didn't change the state to alert and chase because it doesn't need health.



The far sensor of the circle works the same as the sensor of the triangle and the square when it detects a cross or another circle(shape of the same type). However when it detects a triangle or a square it evades.

Near Sensor

The near sensor (small collider) is used to detect shapes and game world elements that are in contact. The squares and the triangles have pretty similar behaviour when they detect another shape in their near sensor. If it is a cross they get health, if it is the same type of shape they recreate and change the state to evade, if it is food they consume it, if they get home they get health and change the state to explore. Lastly in the case that a square detects a triangle or a triangle detects a square they fight. They fight also in the case that they detect a circle.



The circle has the same behaviour as above but it never fights. When it detects a triangle or a square it evades.

Challenges overcome

While I was developing the game a faced two important problem.
1) When a shape detects more than one objects in its far sensor what should it do ? For example if a circle detects a triangle and food in its far sensor should it go to eat the food or should it evade from the triangle. For cases like this one I decided to put some kind of hierarchy on the shapes decision logic. If the shape is already at the state alert and detects another gameobjects it checks what type it is. If the gameobject is hostile (in the case of circle if it detects triangle or square) it evades. However if it is food, same type shape, cross or any other kind of gameObject it does't change its state. In this way the

shape will always reacts to a hostile shape.
2) The second problem I encounter was the case in which the random point for the explore state, is generated on not walkable grid node. As a result the pathfinding algorithm cannot find the shortest path and the shape doesn't move. Also there are some more cases in which the shapes doesn't move. In order to solve this issue I check if the shape doesn't move. If it is still for 60 frames then i change the state to explore and I pick a new random point, so the shape will move.

Design proposals

When I finished my implementation I realise how many of my starting ideas didn't make it to the final game. This is mostly because I didn't have enough time and knowledge to accomplished those features.

Instead of randomly place the food in the food I was thinking to pick some points of the map as fountains of food. In the beginning those fountains will have one food object but over the time the food will expand to the near ground neighbours. In most of the strategy games the resources are found in groups not in individual positions. Also if I was placing the food like that the shapes could remember the position of the food group and go there every time they needed. If all the food of the group was consume the would explore the map to find a new food fountain.

Also i would like to add some ecosystem features. When a shape can find a lot of food it means that it won't have any problem feeding the kids so it will reproduce more. However when it can't find food it won't reproduce because it barelly can feed itself.

In the case of unregulated populations increased I could spread a lethal disease that would kill a big part of the shape population. In order to survive from this disease the shape had to evolve. Shapes that adapted to the new environment would survive, but all the others will die.

Lastly i would like to add some attacking strategies. A triangle won't attack if it just detect an square but it will look at its far sensor for other squares and triangles. If it has the total support of the triangles is greater than the detected squares it will chase the enemy squares, if not it will fall back until it finds more friendly triangles.

Conclusion

I really enjoy developing this simulation game, it help my understand various design concepts like procedurally generated maps, pathfinding, finite state machine AI, and simulation logic. It also help me understand the basis of unity so i can develop more games. I am looking forward to continue this projects by implementing the ideas that I had in mind but I didn't have enough time to implement them in this sort of time.

References

[1] Christine L. Larson; Joel Aronoff; Elizabeth L. Steuer, Simple geometric shapes are implicitly associated with affective value, Motivation and Emotion. 2011

[2] Dr Darryl Charles' Initial Notes on Shapes Design Theme
[3] http://www.roguebasin.com/index.php?title=Cellular_Automata_Method_for_Generating_Random_Cave-Like_Levels
[4] http://www.csharpprogramming.tips/2013/07/Rouge-like-dungeon-generation.html
[5] https://www.youtube.com/playlist?list=PLFt_AvWsXl0eZgMK_DT5_biRkWXftAOf9
[6] https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
[7] https://en.wikipedia.org/wiki/Prim%27s_algorithm
[8] https://en.wikipedia.org/wiki/Kruskal%27s_algorithm
[9] https://www.youtube.com/playlist?list=PLFt_AvWsXl0cq5Umv3pMC9SPnKjfp9eGW
[10] Dr Darryl Charles, week 5, lecture notes, state machines lecture

[11] Mat buckland, Programming game AI by example, 2005