Autonomous Steering Behaviours within a Procedurally Generated City

David Blader 260503611
Marc Jarvis 260500770

COMP 521 Modern Computer Games Final Project
Winter 2015

1. INTRODUCTION

In the third assignment of this course, we explored artificial intelligence within a well-defined space, where a survivor knew the location of all of his objectives and zombie interactions were limited to simple lane changes. In this project, we extend some of these ideas by exploring the use of autonomous steering behaviours to simulate interactions between entities for a more dynamic and life-like effect. We place them within a randomly generated terrain to navigate through with goal locations initially unknown to them. Our main objective is to understand the conditions in which two groups of autonomous entities, zombies and survivors, can achieve their goals given the ability to react to each other using classical steering behaviours as well as share acquired knowledge when grouped together. The distribution of items within the city, the distance between groups of buildings, and the orientation of buildings and entrances will have a strong effect on navigation and survivability.

The implementation uses classical steering behaviours to provide the basic Pursuit, Evade, and Wander actions and a state machine drives the decision making as to which behaviour to use at any given time and which target to interact with. Buildings are instantiated with waypoints to aid in navigation around each building's exterior and in traversing the interior. Knowledge of which buildings have been visited is shared among survivors who come into contact with each other. The city environment is generated by an implementation of a grid based iterated subdivision algorithm. Roads are first constructed according to parameterized constraints and buildings are then constructed on the remaining allotments. Doors and items are randomly distributed amongst each of the buildings.

The results indicate that the layout of the city terrain has a large effect on the survivor's ability to complete the goal and escape the zombies. The amount of survivors and zombies present will have a visible effect on how long it takes for an individual survivor to maneuver around a building accomodate. We found that survivors tend to do best when the city is scaled to have larger road widths and when there are fewer other agents, both fellow survivors and enemy zombies.

In both implementation and sections of this report, David Blader is responsible for the procedural terrain generation and item distribution and Marc Jarvis is responsible for the autonomous behaviour of the entities. A joint effort was made in the integration of both parts and calculating the results.

2. BACKGROUND

Autonomous Movement

Craig Reynolds [Reynolds 1987] explored techniques on aggregating the motion of many individual entities to simulate a natural-looking flocking behaviour. His approach was similar to a particle system [Reeves 1983] such that it does not take any geometric representation into consideration and only considers a single point in space when providing motion over time. Unlike the traditional particle system, the individual entity's state is influenced externally by the state of the other surrounding entities. Originally used as a visual effects element in the film industry [Red3d.com], the simplicity of the flocking model has been generalized and the techniques have been made more accessible for a the gaming community [Reynolds 1999].

Procedural city generation

Procedural generation of cityscapes addresses the problem of the costliness in both time and budget of creating interesting, believable, and detailed cities. As the gaming industry continues to evolve, expectations for high quality environments has been steadily increasing. The traditional approach to obtaining interesting and engaging game environments is to simply increase the number of artists and non-technical designers. This strategy, however, does not yield a proportional amount of content to resources spent on employing artistic talent. (Kelly, McCabe) In George Kelly and Hugh McCabe's "A Survey of Procedural Techniques for City Generation," they present a number of techniques for producing detailed and interesting city environments, through procedural methods developed by work done in the field of computer graphics. These techniques can be employed to generate various city geometries, from a modern grid-like structure, typical of cities like New York's Manhattan borough, concentric patterns found in older European cities, as well as cities efficiently built ontop complex terrains. By implementing one or a combination of these different geometries, large, detailed, and varied city environments can be cheaply produced in a matter of seconds.

3. METHODOLOGY

The objective of a survivor is to explore the city until a radio item has been collected and then reach a helipad location. If multiple survivors come into proximity, they share the use of the radio item and pool their knowledge of which locations have already been explored. The objective of a zombie is to come into contact with a survivor. If a survivor is within view of the zombie, the zombie will communicate that location to any other zombie within its neighbouring region. A secondary item, a torch, exists which will keep zombies at a distance from the survivor when carried. When a survivor has entered a house, he adds it to his memory so that he will not explore it further and this technique will also keep track of previous helipad locations.

Procedural City Generation

The approach we chose for city generation was a simplified, grid based implementation of the iterated subdivision algorithm developed by Nicholas Rudzicz and Clark Verbrugge. A grid based approach was chosen over the more general polygonal strategy detailed in the Rudzciz & Verbrugge paper in part due to time constraints, as well as the Unity environment's insubstantial support for general polygon processing. The algorithm starts by producing a square grid of unit cubes of a size specified by a public parameter, grid size. Once this initial setup is completed, the grid is placed into the 'oversized' stack to be processed by the iterated subdivision method.

Our iterated subdivision implementation is nearly identical from a high level point of view to the version described by Rudzciz & Verbrugge. The oversized stack pops a division of the grid to be bisected along some random point on a random side, so long as both sides are large enough to be divided as determined by the 'max dim' parameter. Once a side has been chosen, a random position between the upper and lower quartiles is chosen to be drawn along. Our algorithm attempts to find a position that will divide the current grid such that it respects the 'min dim' parameter. From there, our implementation introduces a distinguishing feature. The parameter 'road proportion' takes the size of the axis we are drawing along and divides it by the value it's given. We then scale our road by the resulting value. Here is the concept formulized:

$$roadWidth = \frac{currentGridSize}{roadProportion}$$

The concept is inspired by the simple principle that longer, main roads are typically wider than residential roads or alleyways. As a result, longer roads have wider width and shorter roads have shorter widths. Though it is very simple and perhaps somewhat crude, the resulting output has a relatively believable presentation, giving the impression of different neighborhood blocks and alleyways that might be found in a suburb environment.

Eventually the oversized stack will be emptied and the allotment outputs are ready to be built upon. Before setting our buildings, we designate random allotments to be helipads, the end goal which triggers the win state for our survivors. We color them red to make them visually distinct. As we construct our buildings, we try to avoid making buildings that would appear too small, and instead turn them into green, grassy park spaces. Allotments that we deem to be an appropriate size then gain walls and doorways. Before setting our doors, we first get rid of the possibility of doorways which appear to lead off the world grid. The amount of doors for each building is chosen randomly, a number between 1 and the number of available sides as determined by whether the building is built on an edge or corner of the grid. Waypoints are also set at this step for the navigation purposes of the survivor and zombie agents.

Then we place our items. There are two types of items: torches and radios. Torches are the orange items which give a radius on a group of survivors that repels zombies and radios, the black items, are prerequisite for helipad arrival. These items are placed according to probability parameters ‘torch prob’ and ‘radio prob.’ There exists a dependency between these two probabilities, however. Higher torch probability means lower instances of radios, since we only allow one item per building, and if a torch is set, we don’t consider placing a radio. To counteract this, we keep track of whether a radio has been placed or not. If we find that a radio has not been placed, we pick a random building to place a radio in, thus ensuring that the win state is attainable.

Finally, our city is constructed. Once all the pieces have been set, the ‘city scale’ parameter can be applied. This value simply scales the X and Z components of the local scale attribute of the City game object’s transform. Adjusting this parameter has a rather significant effect on how well the agents navigate the city, as characters tend to struggle through narrow alleys, but move rather freely when the city is scaled as such.

Steering Behaviour

The steering behaviours implemented follows closely with code provided by [Buckland 2005] which is directly inspired by Craig Reynold’s work. There are four basic methods which drive the motion of a single entity: Seek, Flee, Arrive, and Wander. Seek provides a steering force directed towards a target’s position and Flee steers in the opposite direction, away from the provided position. Arrive is similar to Seek but with a force proportional to the distance from the target position, which causes the entity to slow down as it gets close. For Wander, a circle is set in front of the entity and then a force is directed towards a random location on this circle’s circumference. This location is updated after a time-out to reduce the jittery-ness of the movement. Seek, Flee, and Arrive are utilized in more complicated behaviours, namely Pursuit, Evade, and Path Following. Pursuit and Evade use Seek and Flee respectively, but take into consideration that their target may be moving and attempts to predict the target’s future location based on its current speed and heading. Path Following assumes that the entity will be moving towards static waypoints and will Seek a target, rather than Pursue. Once a set distance has been reached, the entity will Seek the next target in a waypoint list and if there is only one waypoint left in the list, Arrive is used to bring the entity to a rest.

Group behaviours have corrective forces applied to the above methods and are defined by Cohesion, Separation, and Alignment. For each of these behaviours, an entity iterates over all of the other entities within its field of view (its neighbourhood) and applies an average force. Cohesion calculates the center of mass based on all the neighbouring entity's positions and then Seek is applied to provide a force towards this calculated position. Separation pushes each entity away from each other to keep each other outside of their respective neighbourhoods, which inherently creates a spatial buffer between each entity. Alignment gets the average heading of each neighbouring entity and influences the heading of the current entity to match, so over time each entity moves in approximately the same direction. Group behaviours have been been adapted to be specific to the entity group. For example, survivors with Cohesion enabled are only cohesive to other survivors, to prevent them from grouping with zombies.

Intelligent goal trajectory

The intelligent behaviour of an entity is controlled by a state machine, which switches its steering behaviour based on other entities and waypoints within its view region. Initially, a survivor will Wander until a house enters its neighbourhood. If this house has not been visited before, it enters a new state whereby it looks for the closest waypoint prioritized by 'room', 'door', and then 'corner'. Essentially if it can see straight into the center of the house, it would like to go directly into the room, otherwise it will follow the corners of the house until it reaches a door waypoint and then enters the room. These waypoints are necessary since without these waypoints, the Seek behaviour would constantly force the survivor against a wall in order to get inside the house, rather than go around to find the entrance. Survivors do not collide with the doors and can therefore pass through freely. Since the entity automatically navigates into the center of the room, item pickups are trivially handled by collision detection and the only behaviour that requires implementing is a state change to exit the house. When exiting the house, the survivor chooses an entrance within view and heads towards it. The Evade force prevents a survivor from exiting if there is a zombie on the other side of the door. If all door waypoints are covered by zombies above an integer threshold, the survivor will just wait in hopes that another survivor with a torch pushes the zombies away. Exiting the house simply requires following the door waypoint out and then going back to wander until a new house is found.

The zombie behaviour is also initially set to Wander and once the a survivor is within view, it will Pursue the closest survivor. If a zombie loses line-of-sight from the survivor due to the survivor going behind or inside a house, the zombie will navigate to the waypoints on the house in a similar manner to the survivor's method above, in hopes to traverse around and locate the survivor. If enough time has elapsed without detecting a survivor, the zombie will go back into the Wander state.

When multiple entities of a similar type come into contact, there is both explicit logic applied within the state machine and implicit behaviour based on the steering. When survivors come into contact with each other, Cohesion and Alignment attempt to maintain their proximity with each other. When a survivor is determining if a house has been visited, it consults its own list of previously visited houses as well as the lists of all the other neighbouring survivors, to simulate communication. Only if no survivor within the group has visited a house will they explore it, and the steering keeps them together. As well, if one of the survivors currently has the radio item, then all survivors will search their memory lists for the closest helipad and share that knowledge with the others in the group. Zombie group behaviour also utilizes Cohesion but to simulate less organized behaviour, it does not utilize Alignment. When a survivor is visible to a zombie, it will communicate to the other zombies of his location and then apply a

Seek behaviour to steer towards him. The behaviour of the zombie is kept fairly simple in order to see what inherent behaviours arise out of the automated steering.

4. RESULTS

Both the city generation and entity behaviours are highly parameterized, so we attempted to explore which variables seemed to have the highest impact on the survivor's ability to locate a radio object and then reach the center of the helipad. The city's layout had a large impact, primarily narrowing the streets and increasing the likelihood that the helipad wasn't spawning in the corner of the map. The Wander behaviour can be temperamental and wider streets would often see the survivor turning around before discovering houses on the other side, missing crucial buildings. Some of the failures and deaths that occurred when the zombies were present could have been relaxed if we made the win scenario occur upon reaching any portion of the helipad region, but we opted to be strict and have success defined when the survivor was able to settle into the center of the region.

A trial consisted of automatically generating a city layout based on given parameters and then placing survivors and/or zombies in random locations within the city. From there, the survivors would traverse the space, exploring the buildings and collecting the items while the zombies would wander the streets until a survivor entered their viewing region. Any interactions between entities are handled by the steering behaviours. The first trials were useful in determining what city parameters allowed a single survivor to best navigate the terrain. From there, we increased the amount of survivors to see how their interactions changed an individual's navigation, both with respect to sharing knowledge of buildings already visited and in Cohesive and Alignment forces. Finally, we introduced zombies to further investigate how the behaviours of individuals and groups were altered. Below are the details of our trials:

Parameters used in the following tests:

City constants for each test: Grid size: 50, Minimum dimension: 5, Maximum dimension: 10,
Radio spawn probability: .5, Torch spawn probability: .5

Survivor constants: Max speed = 0.4, Max force = 20,
Wander radius = 1.5, Distance =1, Jitter = 0.65, Wander update = 3 seconds

Zombie constants: Max speed = .2, Max force = 20

Test A city parameters: City scale = 6, Road proportion: 8, Number of helipads: 7

Test B city parameters: City scale = 5, Road proportion: 12, Number of helipads: 5

Test C city parameters: City scale = 5, Road proportion: 20, Number of helipads: 5

There are a few common types of failures observed, which prevent the survivor from completing the objective:

Timeout : A basic time-out, where the survivor is still navigating the world, but does not complete the objective within an allotted time (roughly 240 seconds).

Stuck : An anomaly where the survivor attempts to reach a waypoint that shouldn't be accessible and gets caught in a corner.

Oscillate: An anomaly where the survivor doesn't maintain his memory of visited waypoints properly and oscillates back and forth. This is only present when a building has only one door and the survivor is traversing its backside and cannot view any of the doors.

Killed: The survivor came into contact with a zombie.

Time taken for survivors to complete the objective with no zombies present..

Trials	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Test A	8.848	14.538	41.993	153.504	12.243	46.020	35.598	39.981	35.186	20.420
Test B	167.282	16.615	Stuck	Stuck	Stuck	35.344	Oscillate	23.369	86.597	84.692
Test C	96.002	Oscillate	31.959	86.471	114.038	166.701	9.167	Stuck	Oscillate	120.726

Time taken for survivors to complete the objective with multiple survivors and no zombies.

Trials	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Test A: Successes of 5 survivors	4 Success, 1 Oscillate	4 Success, 1 Timeout	2 Success, 2 Stuck, 1 Oscillate	2 Success, 2 Timeout, 1 Stuck	4 Success, 2 Timeout, 1 Stuck	2 Success, 3 Oscillate	2 Success, 1 Timeout, 1 Stuck, 1 Oscillate	4 Success, 1 Oscillate
Times of the successful survivors	49.85 135.1 34.38 361.89	16.97 47.16 47.72 113.90	47.882 102.178	135.992 213.147	54.620 55.726 125.90 157.667	197.146 346.486	10.032 11.043	46.032 72.169 136.115 147.163
Test A: Successes of 10 survivors: without Cohesion	4 Success, 6 Timeout	6 Success, 2 Timeout, 2 Oscillate	2 Success, 8 Timeout	7 Success, 1 Timeout, 2 Stuck				
Times of the successful survivors	18.861, 43.629, 61.836, 185.855	34.716, 54.823, 60.643, 68.164, 112.816, 145.026	67.868, 196.35	11.664, 18.542, 35.471, 57.683. 69.279, 77.374, 83.082				
Test A: Successes of 10 survivors: with Cohesion	7 Success, 2 Timeout, 1 Stuck	4 Success, 4 Timeout, 2 Oscillate	5 Success, 5 Timeout	6 Success, 4 Timeout				
Times of the successful survivors	12.501, 27.479, 27.877. 31.255, 43.566, 60.819, 95.015	105.808, 119.282, 125.393, 155.469	60.123, 96.693, 98.483, 104.339, 159.066	24.606, 29.849, 39.192, 46.315, 172.504, 180.459				

Time taken for survivors to complete the objective with variable zombies present using Test A settings.

Trials	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1 survivor, 5 zombies	132.182	157.659	106.619	68.763	135.199	49.424	Stuck	64.798
1 survivor, 25 zombies	54.527	203.286	159.675	killed	killed	Stuck	46.945	129.547

Trials	1	2	3	4	5	6	7	8
1 survivor, 50 zombies	killed	38.92	killed	killed	killed	killed	killed	killed
10 survivors, 50 zombies	7 killed, 3 Timeout	2 Success, 4 killed 4 Timeout	1 Success, 7 killed 2 Timeout	8 killed, 2 Timeout				
Times of the successful survivors		45.309, 120.298	124.268					

The use of Cohesion had interesting results. One one hand, if two survivors within proximity were following waypoints, sometimes the pull on each other would not allow them to get to their waypoint and they would remain still with their net-zero velocities until an additional force disrupted the equilibrium. This would slow down the survivor and lengthen the time to complete the goal. On the other hand, the forced proximity resulted in many survivors completing the goal within a small time-frame from each other, which gave the appearance of teamwork. Also, even though 5 zombies did not pose much of a threat at all to the survivor, they generally increased the time for success because of the extra time spent Evading and then returning to the house originally intended to explore. An observation not shown in the above charts is that when the zombies were killing the survivor, it was mostly due to the survivor being cornered by multiple zombies against a wall. The torch was never active on a survivor when he was killed.

5. Conclusions / Future Work

The city generation algorithm can be extended in a number of ways to improve realism. In general, our project could be extended to have different types of city geometries similar to those detailed in the Kelly & McCabe article. Other examples more specific to the work completed for this project would be a snapping function that connects nearby but disjoint roads. We had also intended to include different types of doors, both metal and wood, which can be seen in our implementation. The difference between these two types of doors being that wood doors are destructible by zombies. Unfortunately, this functionality did not get fleshed out to its full potential. At the early design phase we had hoped to have varying building types, each with their own probability of spawning special items. Implementing varied building architectures and exteriors would add a great amount of visual impressiveness to the generated cities, and a wider array of item types could have interesting implications on both survivor and zombie behaviours. There are near endless possibilities for including more detail and realism to the city. If made sufficiently interesting and realistic, it could be used in a small-scale game to produce new environments with each play through, as today's gaming industry typically pushes products that have high "replay value." We could also include interactive tools to allow manipulation of the city output for designers to make their own.

As for the steering behaviour, there can be many improvements in how the collisions are detected, as it is currently only using the Obstacle Avoidance method described by [Reynolds 1999]. Steering behaviours take time to find the correct weighting in each situation to get the desired effects and "force wells" naturally pop up, causing an entity to remain static or stuck. The use of behaviour trees may provide an interesting alternative to the state machine when determining behaviour actions.

The mixture of autonomous behaviour within a procedural world can benefit from a designer's touch, but it's the random nature of the outcome which provides the most interest to the viewer.

6. REFERENCES

[1] Reynolds, C. W. (1987) Flocks, Herds, and Schools: A Distributed Behavioral Model, in Computer Graphics, 21(4) (SIGGRAPH ’87 Conference Proceedings) pages 25-34.

[2] Reeves, W., T., "Particle Systems--A Technique for Modeling a Class of Fuzzy Objects," acm Transactions on Graphics, V2 #2, April 1983, and reprinted in Computer Graphics, V17 #3, July 1983, (acm S[GGRAPH '83 Proceedings), pp. 359-376.

[3] Red3d.com,. 'Behavioral Animation'. N.p., 2015. Web. 13 Apr. 2015. (http://www.red3d.com/cwr/behave.html)

[4] Reynolds, C. W. (1999) Steering Behaviors For Autonomous Characters, in the proceedings of Game Developers Conference 1999 held in San Jose, California. Miller Freeman Game Group, San Francisco, California. Pages 763-782.

[5] Buckland, Mat. *Programming Game AI By Example*. Plano, Tex.: Wordware Pub., 2005. Print.

[6] Unity3d.com,. 'Unity - Game Engine'. N.p., 2015. Web. 13 Apr. 2015.

[7] Rudzicz, Nicholas, and Clark Verbrugge. An Iterated Subdivision Algorithm For Procedural Road Plan Generation. Gram.cs.mcgill.ca. N.p., n.d. Web.

[8] Kelly, George, and Hugh McCabe. "A Survey of Procedural Techniques for City Generation." ITB Journal (2006): 1-15. Web.