Slide 1

Tile-Based Game Development Research and Backbone Coding

* Research began with a paper on 2D game development and tile-based game development.
* Project based upon a tutorial series focused on tile-based 2D game design in Unity, available via GitHub <https://github.com/TeamPorcupine/ProjectPorcupine>
* The main backbone design implementation for my project uses the Component structure of Unity to allow a separation between the visuals in the engine, and the code behind all the logic.
* Standard C# classes are used to apply the logic of the game, and a few simple scripts derived from Monobehaviour are used to link the GameObjects to the standard C# classes. The link is made using Actions and callbacks. Whenever a piece of furniture needs changing, the action is triggered and then functions are run that are needed to adjust the GameObjects.
* In the beginning there were only 3 of these Monobehaviour classes; World Controller, Mouse Controller, and Furniture Sprite Controller.
* The World Controller uses a singleton system to keep the rest of the code together. The mouse controller is used to deal with all of the inputs, currently this is just the mouse. The Furniture Sprite Controller works with the Mouse Controller to allow furniture to be placed into the game world at certain tiles, and will be responsible for updating the furniture GameObjects if they need updating.
* Other standard C# classes were created including: World, Tile, Furniture, and Furniture Actions.
* The World class is responsible for keeping all of the game’s information such as what furniture is in the game, what characters are in the game, all of the possible furniture that can be placed into the world, and all the other overseeing of the game environment. The Tile and Furniture classes act as templates and represent all the Tiles and Furniture in the game respectively.
* Furniture Actions is a unique class. It is a static class that deals with the ‘update functions’ of the furniture. Most furniture does not need its own update function; this class contains a few functions that functions are linked to when their prototypes are created in World. At this point the only furniture that needs an update function is the Door, which will open over time if a character wants to walk through it.

Slide 2

* The best sources I found for character development research is looking at popular and well-designed games with sophisticated AI. I used the RimWorld as the main source due to the similarities with my project. I looked into the skills, backstories, traits, mood and thoughts. I used these to develop my characters. I then began coding and I created two more scripts. The Character class is not derived from Monobehaviour and is used as the template for all characters in the game. The other script is the Character Sprite Controller and is derived from Monobehaviour and acts the same as the Furniture Sprite Controller. It is in charge of keeping track of the character GameObjects and updating them when they need to be.

Slide 3

* Good pathfinding is such an important part of good AI and so good research needs to be done to create the best pathfinding that is possible
* Research was done on Hybrid Pathfinding, direction based heuristic pathfinding, multigoal pathfinding, and pathfinding for partially explored game environments.
* The hybrid pathfinding is used when there are different pathfinding techniques used in a dynamic environment, such as long-distance travel and short-distance enemy avoidance. The paper talks about the best way to integrate them both, swap between them and even use them both at the same time to create very realistic looking and smart AI. This isn’t needed much in my project due to short-distance enemy avoidance behaviour not being relevant.
* The direction based heuristic pathfinding looked at A\* pathfinding for a quick and reliable algorithm. It talked about the A\* algorithm in depth and then went on to talk about heuristics in general.
* The multigoal pathfinding was research done on working out if knowing that a character needs to go to multiple destinations at the start of the pathfinding search and using that to change the route they take will actually help them and speed up the time to get to every goal one after the other. It used and explained a few well developed techniques.
* The partially explored game environment pathfinding was a paper focused on the development of AI that can “see” its environment and path finds based upon its knowledge of the environment, instead of assuming the character has 100% map awareness, and as the character moves and discovers more about the environment, it will actively update its pathfinding. This paper interested me greatly as this kind of pathfinding sounds like a very realistic technique and a promising one for this project, it does however have a flaw, in that the character needs additional development so it can “remember” the environment, and adjust its pathfinding as it learns.
* I finally decided to go with A\* as the first implementation of the pathfinding, with the partial explored technique in the back of my mind as a possible future develop step.

Slide 4

* The pathfinding implementation began with adding a Node template class and an Edge template class. These are used to create a Tile Graph of all the tiles that are linked together. The tile graph is then used within the A\* algorithm to work the best path to the destination.
* Every piece of furniture has a movement cost, this means that character will go around the furniture if they can but if it is better to go over the furniture then they will.
* The character uses the route by asking for the next tile in the list of tiles, they then move to that tile by lerping from the current tile to the next tile, and the character sprite controller updates the position of the GameObject. Before moving to the next tile, the character checks to see if the tile is enterable, this is important because if the next tile has a door, the character needs to wait, and the door is notified that it needs to open. Then once it is open, the enterability of the tile is changed to open, and the character carries on moving. If the next tile has a movement above 1, the character movement speed will be adjusted while moving through that tile, and then goes back to its normal speed.
* A button was added to the Unity project which sets the mouse mode which allows clicking on a tile set the character’s destination to that tile. That was used to test the algorithm and fix anything that was broken.

Slide 5

* Research was based on papers focused on NPCs and advanced AI.
* A paper was found on emotion based synthetic characters. It explained about how giving NPCs emotion that can dynamically adjust to what the player, or other NPCs, do can create very realistic and advanced AI behaviour. The model uses fuzzy logic, and memory to achieve this. This is something that is being planned to be added into my character AI system.
* The next paper was about designing human-like NPCs using pheromone maps. It talks about the character learning about its environment and adjusting its behaviour based on certain things, such as the amount of enemy movement in a certain area. I did not find the paper very relevant for this project but will be looked back at if a line of sight and environment learning behaviour is added to the characters.
* The next paper was adaptive behaviour control models for NPCs. It talks about fuzzy logic and its advantages of disadvantages. The main bulk of the paper was not relevant but the fuzzy logic section will be looked back at if fuzzy logic is something that gets implemented into the AI.
* The next paper was about component based hierarchical state machines. It talked about a very advanced form of finite state machines and how they can be used and why they may be better or worse than simpler forms. I felt like the techniques they were describing were unnecessarily complicated for this project and allowed a lot of advanced AI techniques that will not benefit the realism needed for this project.
* The final piece of research I used is a book from the library and general game coding. The section on AI was very thorough and explained a lot of ways to improve AI realism. Lots and lots of notes were taken from this section but the main points were finite-state-machines, decision trees and fuzzy logic. I used this book, more specifically the finite state machines section, to begin the next task of the project which is the employee AI code.

Slide 6

* First step into the employee coding was changing the character class to be abstract. All characters are either employees, or customers, and so the character class will not need to be instantiated without an employee or customer class also being instantiated.
* The employee class was created and inherits from the character class.
* A job class was also created which keeps track of all of the states that are used within the finite state machine. The system has primary states and secondary states. The primary states currently are:
  + Serve On Checkout
  + Work Stockcage
  + Empty Stockcage
  + Work Back Stock
  + Face Up
  + Count Checkout Money
* The secondary states currently are:
  + Go To
  + Go With
  + Idle
  + Use
* The primary states represent the name of the entire job an employee needs to do, and the secondary states represent individual tasks that are required in order to do the entire job.
* Initially I implemented the finite state machine into the Job class, but this ran into a large amount of problems, not to mention is doesn’t make sense that the job itself is deciding what needs to be done. It makes more sense for the Job class to simply hold information about the job and the employee makes the decision about what they need to do next. Because of this, the finite state machine was moved into the employee class and is a part of the Update system.
* Each Job has a tile that the employee needs to be at in order to complete the job. So when an employee gets a new job, they work out if they are at the tile they need to be at to perform the job, if they are not, they will move there. Next they will Use the furniture they need in order to do the job. Sometimes an employee with be idle, this means they have a job, but they cannot currently do it. This happens the most when an employee’s job is to man a checkout, but there are no customers in the queue.
* All character have an Update\_DoThink() function that is overridden for the employee class, and in the future the customer class. Inside this is the finite-state machine. For the employees, another update function called Update\_DoJob() also runs after the Update\_DoThink() function which contains the logic of what the employee will do for the job, such as scanning items, or moving them from a trolley to a shelf.

Slide 7

* After this I began thinking about the beta testing and that will be looked for in the AI. At this stage it was hard to find specifics but the general areas for testing are:
  + Job priority decisions
  + Communication
  + Traits, thoughts and moods
  + Line-of-Sight if implemented.
* Things that the testers should not pay attention to are game balancing, graphics and sounds because these are not things that the project cares about.
* Next I implemented Stock into the game. Stock is a class and represents a template for all stock in the game. All furniture can have stock on it, however the stock cannot exceed the weight limit of the furniture, each item of stock has a weight attached to it. Walls have a weight limit of zero, this means that no stock can be added to it. Furniture and character classes have new functions that allow movement of stock between them. Most of the function return a bool, so that if the movement was unsuccessful, either because the stock was too heavy or it wasn’t present in the first place, then the other entity knows it didn’t receive the item and can adjust its wants accordingly.
* When testing the employee checkout logic, employees were picking up stock, scanning it, putting it back down, then picking the same item back up again. This was fixed by adding a ‘sold’ variable to all stock so that employees know if it has already been scanned and does not need to be scanned again.
* The next task I completed was adding selection UI to the Unity project. This did not advance the AI, however, it is now easier to debug and track what items are on what furniture, and whether a piece of furniture is being used by a character.
* After this I added all furniture into the game required for this project, along with all stock required for this project, and then create a shop environment that can be used for all testing from this point onwards. This required more logic coding, such as furniture rotation.

Slide 8

* After this the next stage was to add movable furniture into the game. Two of the furniture can be moved, the trolley and the stock cage. Adding movable furniture proved very complicated due to the initial furniture code not working correctly as once a piece of furniture was added onto a tile, it was not meant to be moved. The furniture actions class helped with this by allowing movable furniture to adjust its position in a very similar way to how the characters move around the world. However, the furniture does not move by itself, a character is needed to tell the furniture where to go next. I am very happy with this implementation as it is very realistic and a good addition to the AI logic.
* The next code to be added was character code that allowed characters to move movable furniture if it is in their way.

Slide 9

* Initially they simply pushing the furniture until it was out of the way, but this quickly didn’t work and a better implementation was required. Now, a floodfill is created from the furniture’s tile, and it finds the first empty tile that isn’t in the character’s path to their goal, and then the character moves the furniture there.
* After testing this a few times another problem was found. Characters could trap themselves by repeating this process again and again which is no good. The code was reviewed and now a function is called recursively which checks the tiles where the character and furniture will be eventually and seeing if this continued path will end up trapping the character. If it does, the initial valid tile is set to invalid, and the floodfill repeats, but now the originally good tile, is not found, and this means the character will not get trapped.

Slide 10

* Despite the employee AI needing to be done by 20th January, it has not been completed. This was due to the Stock implementation and the movable furniture implementation being much more complicated and longer to create than anticipated.
* However, this will allow the rest of the AI coding to be completed quicker than anticipated from this point onwards due to it needing to be less complicated now that the movable furniture has been created in such an advanced way. Over the next week, the employee AI will be completed and employees will move around the shop completed the tasks required.
* Interaction between characters will be done during the customers coding phase, as the customer coding phrase will take less time than given on the Gantt Chart due to its logic being so similar to the employee logic, and the inclusion of the character abstract class allowing character logic as a whole be implemented for both type of character at the same time.

Slide 11

* Here is the initial plan for the project.
* The first part, pathfinding, has been implemented successfully and works very well with job, and movable furniture.
* The second part, Job Decisions, has been mostly implemented. The finite state machine is in place, and the logic is ready to be coded. The employees can correctly decide what job they need to do, but they just don’t yet know what to do once they are at the job location.
* The third part, the shopping choices, has not been implemented and that will begin this week, or next week, once the employee AI is finished.