**Whose a good boy. Dog walking sim**



Player controlled character can pick out a dog from a house. Then take the dog for a walk. The game is complete once the dog has become tired and has been returned home.

The world will have a maze, ducks, squirrels, rabbits, balls, treats, trees and other dogs. Differnt breeds of dog will have differnt AI and personalities and interact with the world around them differently, differnt breeds will become tired by doing differnt things.

eg. grey hounds will like to chase rabbits, terriers chase squirrels, pugs dont chase anything, labrador likes to chase balls, pitbulls will attack other dogs, some dogs like to play with others. some dogs will be obiedent others will be easy to distract

**Milestone 1: Create moving npc**

-NPC can randomly move around map. - done

-NPC can move around map in a general direction - done

-implement walls - done

-implement roads

-implement csv file map reader

**Milestone 2: Basic path finding**

-NPC can move from one point to another around walls - done

**Milestone 3: basic NPC and player interaction**

-npc will come to player when called - done

-npc will fallow player when called - done

-npc will stay when told to - done

-throw ball. Npc will run to ball and return it to player

-npc will run away from player maintaining certain distance - done

-npc will run away from player towards object tree.

-aggressive npc will run after non aggressive npc

**Milestone 4: Advanced Pathfinding**

-NPCs can pathfind in diagonal direction

-NPCs will make most efficient path based on terrain

-NPC have vision distance - done

-npc has hearing distance - done

**Milestone 4: Advanced NPC and player interaction**

-create basic animal NPC

-create dog npc

-dog and player can play fetch

-create squirrel NPC

-dog and player collision control

-player can pat dog

**Milestone 5: NPC and NPC interaction**

-dogs can chase and kill ducks

-ducks will run away from dogs

-dogs can play together

**Milestone 6: implement pygame graphics.**

-implemt ground sprite - done

-implemnt barrier class (tree, endofmap, water)

-implement barrier sprites - done

-implement player sprite - done

-implement dog - done

-implement duck sprite

-player can pat dog and a heart appears.

-implement death sprite

**Milestone 7: implement basic game loop**

-dog can become tired from playing fetch

-game can be won by returning tired dog home

**Milestone 8: implement basic sounds**

-dog can bark

-dog can growl

-duck can quack

**Milestone 9: Create Game World**

-pavement ground

-create road ground

-create tree barrier

-create rabbit hole ground

-create house barrier

-create dog house Barrier

-create map

-create maze

**Milestone 10: Advanced Pathfinding**

-NPCs can pathfind in diagonal direction

-NPCs will make most efficient path based on terrain

-NPC have vision distance - done

-npc has hearing distance - done

**Milestone 11: create animal npcs**

-create rabbit

-create duck

-create greyhound

-create border collie

-create terrier

-pitbull

-create pug

**Milestone 12: advanced NPC and NPC interaction**

-dogs can chase rabbits

-dogs can kill other dogs

-dogs can chase ducks

**Milestone 13: animation**

**Milestone 14: implement fun stuff**

-satan dog

-shot gun

-other people

-german shepherd

-cars

**AI**

**Pathfinding Algorithsm**

**finite state machine**

Animal behavoir and AI is controlled by a finite state machine.

below is an example of super basic state machine for coin operated turnstyle



the NPC finite state machines will be changed by both direct input from player and also interactions with enviroment influencing the state animals are in.

eg. player presses space - dog will change from stay to follow

when dog goes close to duck the duck will go from stay to flee state

**data structes**

Game grid is a 2d array the stores if a node is a barrier or empty. this data structure is used by the path finding algorithm to find efficient paths from on point (start) in the game grid to another (end). this grid does not store anything about the player or npc character only if a position on the grid has a barrier or not.

below is how the game grid stores the nodes

[

[barrier,barrier, barrier,barrier,barrier, barrier],

[barrier, start, empty, empty, empty, barrier],

[barrier, empty, empty, empty, empty, barrier],

[barreir, barrier, barrier,barrier,empty, barrier],

[barrier, empty, end, empty, empty, barrier],

[barreir, barrier, barrier,barrier,barrier, barrier]

]

the algorith will review the nodes and find an effienct path from start to end through empty nodes as visualised below



once an efficent path is found the algorithm returns the following which can be used by npc for pathfinding

[end, step 8, step 7, step 6, step 5, step 4, step 3, step 2, step 1, start]

improvements to the algorith will add weights for terrain and diagonal movement. at the moment all nodes have the same weight. this improvement will allow npcs to move diagonally and choose to avoid rough ground in prefernce for paved roads.

**nodes and blocks**

**NPC classes**

**Animal**



states

stay: animal does nothing

flee: animal will do nothing, unless player is close then the animal will move away

follow: animal will continue to go to the player until they are 5 square close

**Dog**



the dog has the following states

States:

stay: dog does nothing

follow: dog goes to player

follow sniff: dog will move to nodes randomly, providing nodes are not too far from player

flee: dog will move to node that is far enough away from player

sniff

flee sniff: dog will randomly move around to nodes not too close to player

**Squirrel**



sniff: squirrel will move randomly around but wont move to any node too far away from a tree

Go towards tree: squirrel will move to node that is closer to a tree and further away from player

hide in tree: squirrel will stay on tree node and do nothing