Design Work
(Sebastian Pelka, Filip Gutica, Sanders Lee)

Table of Contents

Abstract

This document presents the collaborative efforts of our team's contribution to the project. Preliminary design of our project includes:

- Artificial Intelligence
- Player Characters
- Non-Player Characters (Gatekeepers)

Module: Artificial Intelligence

Author: Sebastian Pelka

I have found three interesting AI archetypes on the internet: The "Guard Dog", the "Patroller", and the "predator", which are common AI types found in many games nowadays. These archetypes can each be divided into a ranged or melee flavour.

I believe that it is in the best interest of the game to implement a few different AI modes, to make the minions that populate the map a little more interesting, and also to keep all players engaged.

The Guard Dog:

The guard dog is an enemy archetype which will be in a WAITING state at its spawn point. It will be aware of entities in cells of a given radius from its spawn point (a guard area). When a PC enters its guard area, the guard dog goes into a MOVING state and @ for the offending PC. When the guard dog is in combat range of the PC, it will go into the ATTACKING state. If the guard dog hit points are reduced to zero, it goes into a DEAD state, and is removed from the game. If the offending PC is killed, the guard dog will check for another PC to engage. If it fails to find a PC, it will go into a MOVING state to return to its spawn point. Upon arrival, it enters a WAITING state.

The Patroller:

The patroller is an enemy archetype, which will be in a MOVING state normally. It will have n nodes, and will travel from node to node, in some circuit. The patroller searches for a PC and is aware of the cells nearby it as it moves around the map. When a patroller becomes aware of a PC, it will break from its patrol to pursue any PC that it sees, until the PC leaves the cells that the patroller can see, or until either the PC or the patroller die. If the PC is killed, the patroller will return to its patrol, which involves checking its nearby cells for new PC targets.

The Hunter:

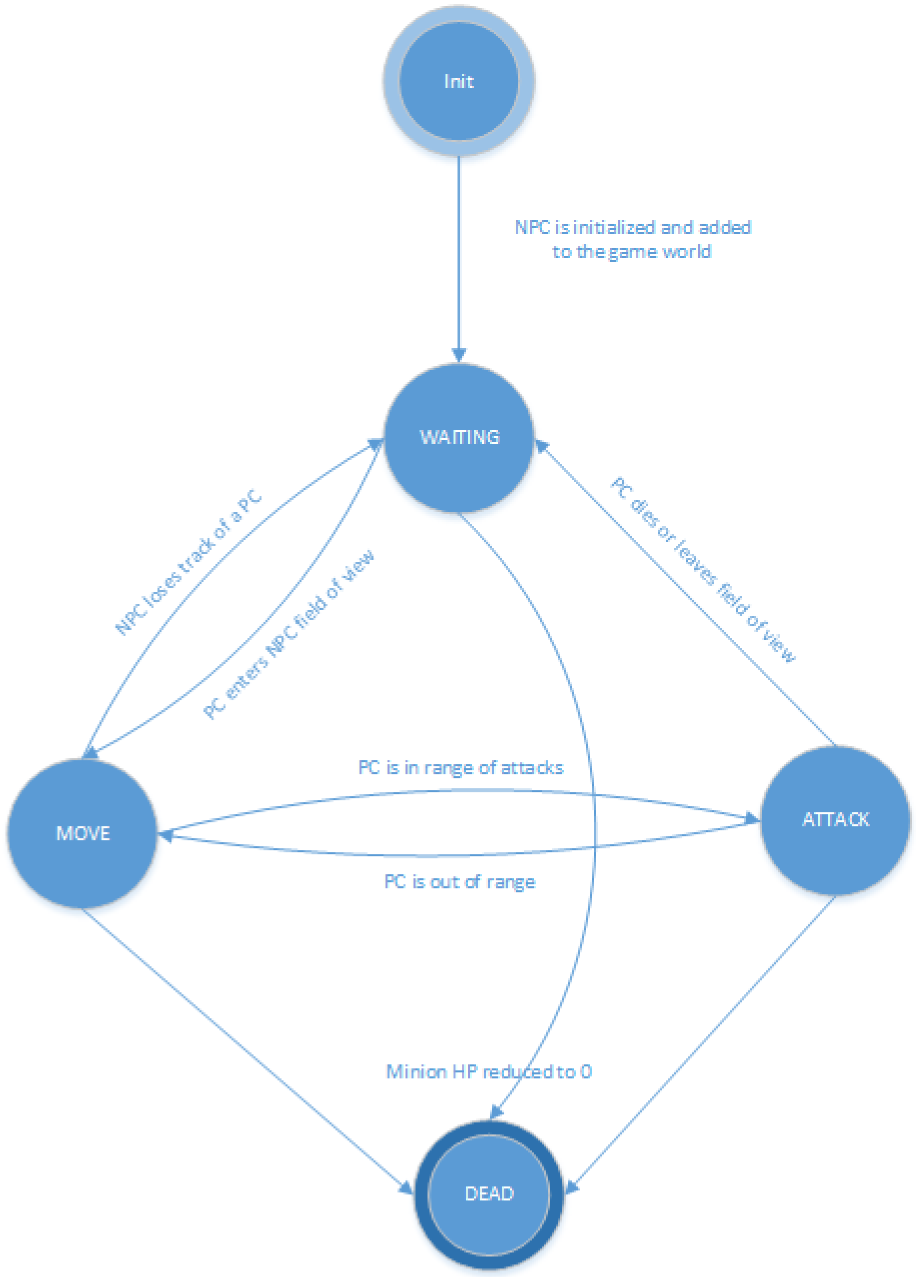
The hunter is an enemy archetype which will at the beginning of a game, select a PC as a target, and immediately move towards the PC, with the intent to engage them relentlessly. The hunter will have a WAITING state, for target selection purposes, but then should spend the majority of its time in a MOVING or ATTACKING state. Upon killing its target, the predator will go into a WAITING state and select a new PC to attack, at which point it will go into a MOVING state once more.

The Sentinel:

The sentinel enemy archetype is a more specialized version of the "Guard dog" mentioned above. It begins in a WAITING state, and monitors the cells surrounding its spawn point. Should a PC enter this guard area, the sentinel will enter the ATTACKING phase and take action against the NPC (spawning smaller minions, like a beehive spawning bees). If the PC leaves its guard area or is destroyed, the sentinel will reset to a WAITING state.

General AI State Diagram

Author: Sebastian Pelka


Init
NPC is initialized and added to the game world
WAITING
NPC loses track of a PC
PC enters NPC field of view
PC dies or leaves field of view
PC is in range of attacks
MOVE
ATTACK
PC is out of range
Minion HP reduced to 0
DEAD

INIT: In this state, the NPC is initialized and added to the game world. Once this has been accomplished and the NPC object has been added to the game world, it enters a WAITING state.

WAITING/IDLE: In this state, the minion will be carrying out its default behaviour (moving from node to node, or holding at a location), while checking the cells in it can see for any PCs. When a PC is detected, the minion changes to the MOVE state. If a minion is killed (set to 0 HP) without detecting a PC, it changes to the DEAD state.

If a minion is not at its start location/spawn point, and there are no PCs in range, it will go into the MOVE state and return to its start location.

If a minion is attacked it will go to the MOVE state, towards its attacker.

MOVE: In this state, the minion will move towards a PC until they are in range of their primary attack and enter into the ATTACK state.

If a minion is not at its starting location, and there are no valid targets in its guard area, it will reset to the WAITING state.

If a minion hit points are set to 0 or less while in this state, it changes to the DEAD state.

ATTACK: When a minion is within its weapon's firing range, it will begin attacking the PC until

- the PC moves out of range of the attack, at which point the minion will go to a MOVE state.
- The minion is set to 0 or less HP, in which case it goes to the DEAD state.
- The PC is killed, in which case the minion goes to the WAITING state.
- the PC leaves the guard area, in which case the minion goes to the WAITING state.

DEAD: When the minion is in this state, its data is removed from the game. (whatever that might mean; I'm not sure myself.).

Pseudo code: “Guard Dog”

Author: Sebastian Pelka

```
INIT
{
   initialize the monster object
   add the monster to the game world
   go into the waiting state
}

WAITING
{
     for every n frame(s) of the game
          {
          if the monster has less than 1 hit point
               {
               go to the DEAD state (remove the monster object from the game)
          }
          check all cells within the field of view of the spawn point in sequence
          if a PC is detected in one of these cells
          {
               if the PC is not already in the list of target PCs, add it to the list
          }
          Loop through the list of target PCs
          {
               if the PC is outside the field of view, remove it from the list
          }
          If at least one PC is in the monster’s target list
          {
               go into the MOVE state (to move to the target PC location)
          }
          if there are no PCs in the monster’s target list
          {
               go into the MOVE state (to move back to the spawn point)
          }
     }
}
```

MOVE

```
{
    for every n frame(s) of the game
    {
        if the monster has less than 1 hit point
        {
            go to the DEAD state (remove the monster object from the game)
        }
        update the locations of the first PC on the list of target PCs
        if the PC is dead
        {
            return to the WAITING state
        }
        determine if the PC is in the monster’s attack range
        if the PC is not in the monster’s attack range
        {
            move towards the PC
        }
        else, go into the ATTACK state
    }
}
```

```
ATTACK
{
    for every n frame(s) of the game
    {
        if the monster has less than 1 hit point
        {
            go to the DEAD state (remove the monster object from the game)
        }
        if the target PC is dead or is outside the field of view
        {
            go to the WAITING state (to scan for new targets)
        }
        if the target PC is in the monster’s attack range
        {
            attack the target
        }
        else
        {
            go to the MOVE state (to move into attack range)
        }
    }
}
```

```
DEAD
{
    remove the monster from the game world
    de-allocate memory
}
```

Module: Player Characters

Author: Sanders Lee

Controls

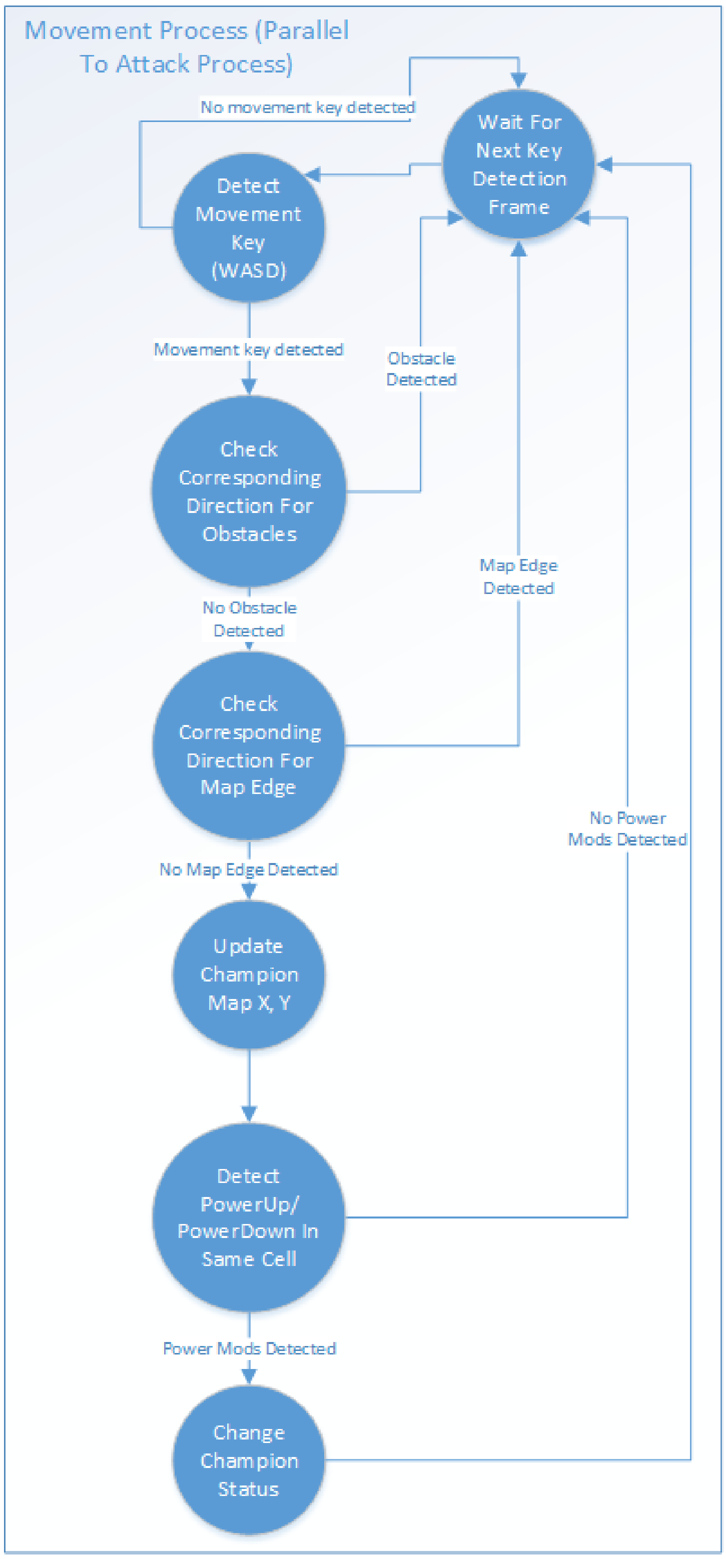
- Movement
  - WASD keys
  - Collision detection, i.e. can't move into an occupied cell
  - No moving off the map
- Attack
  - L Mouse = normal attack
  - Mouse wheel = toggle between melee and long-range normal attack
  - R Mouse = special ability: short cooldown
  - L Shift + R Mouse = special ability: medium cooldown
  - Spacebar + R Mouse = special ability: long cooldown
  - Cooldowns

State

- X, Y position on map
- HP
  - Dead @ 0 HP
- EXP/Level
- Stat boosts
- Status effects, i.e. blessings & curses from deities
- Game points
  - Variable bonuses depending on task completion, examples include:
    - Beat a minion
    - Beat a mini-boss
    - Beat a boss
    - Beat a champion
    - Last survivor
  - Recording points

Champion Movement State Diagram

Author: Sanders Lee


Movement Process (Parallel To Attack Process)
Wait For Next Key Detection Frame
No movement key detected
Detect Movement Key (WASD)
Movement key detected
Obstacle Detected
Check Corresponding Direction For Obstacles
Map Edge Detected
No Obstacle Detected
Check Corresponding Direction For Map Edge
No Power Mods Detected
No Map Edge Detected
Update Champion Map X, Y
Detect PowerUp/ PowerDown In Same Cell
Power Mods Detected
Change Champion Status

Wait For Next Key Detection Frame: Assume that there's a small delay until the game would take in keyboard input so users can't use bots to spam control input.

Detect Movement Key: A time frame in which the game will take *one* key from WASD as a valid movement input

Check Corresponding Direction For Obstacles: self-explanatory, can't move in a direction with solid obstacles

Check Corresponding Direction For Map Edge: can't move off the edge of the map

Update Champion Map X, Y: if there are no problems, move champion into the corresponding map tile

Detect PowerUp/PowerDown In Same Cell: see if there are champion upgrades/downgrades in the cell the champion just moved into and absorb it

Change Champion Status: change champion status according to the upgrade/downgrade detected, <u>we have no idea what these could be at this point in time</u>

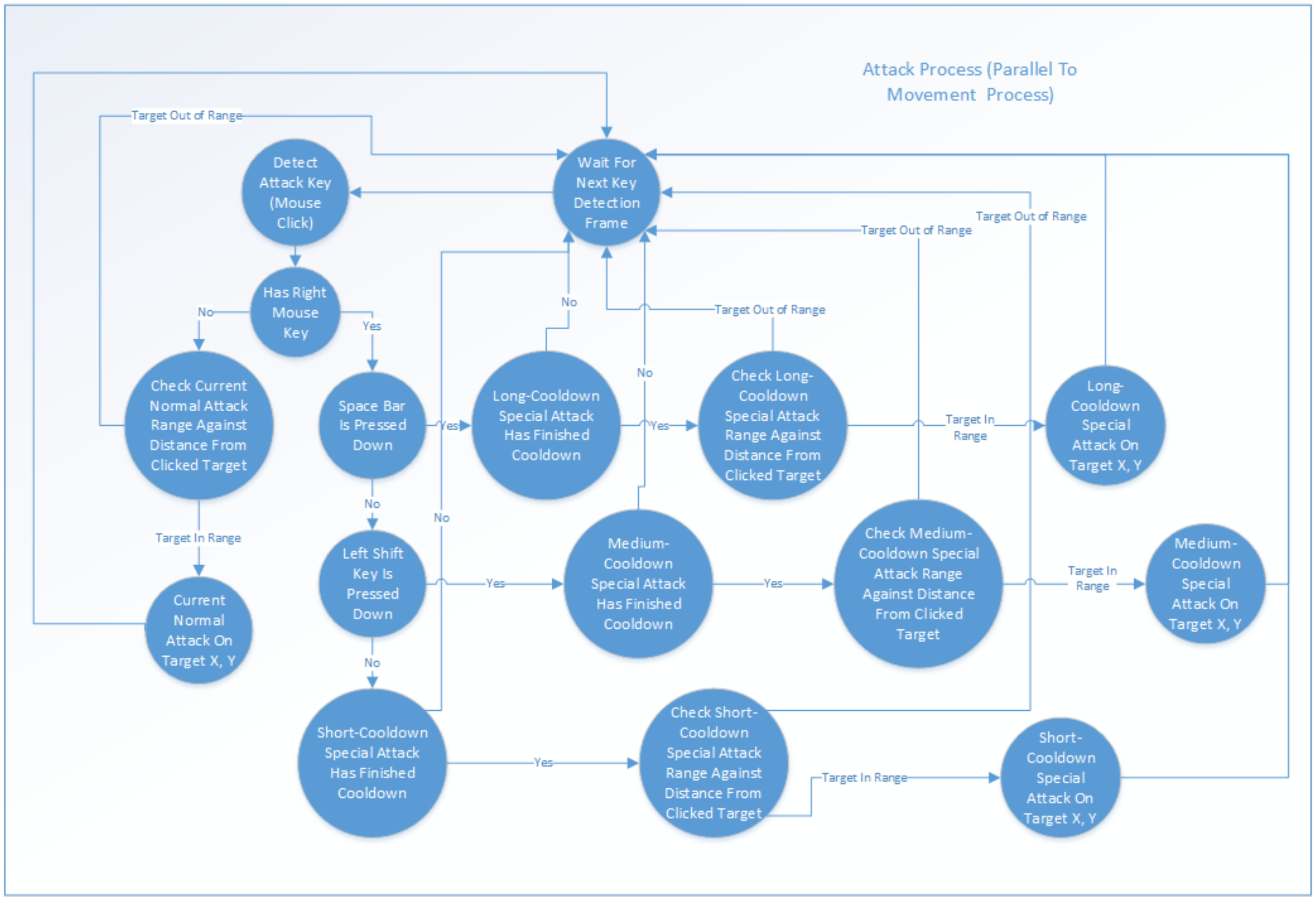
Movement Pseudocode

Author: Sanders Lee

```
function to detect movement
  if WASD key is pressed
    case W, if "up" has no obstacles, if "up" is not the edge of the map,
      then move champion "up"
    case A, if "left" has no obstacles, if "left" is not the edge of the map,
      then move champion "left"
    case S, if "down" has no obstacles, if "down" is not the edge of the map,
      then move champion "down"
    case D, if "right" has no obstacles, if "right" is not the edge of the map,
      then move champion "right"
    for all cases,
      update champion X, Y on map
      if powerup/powerdown exists in current cell
        update champion status
```

Champion Attack State Diagram

Author: Sanders Lee


Attack Process (Parallel To Movement Process)
Wait For Next Key Detection Frame
Detect Attack Key (Mouse Click)
Has Right Mouse Key
No
Yes
Check Current Normal Attack Range Against Distance From Clicked Target
Target Out of Range
Target In Range
Current Normal Attack On Target X, Y
Space Bar Is Pressed Down
Left Shift Key Is Pressed Down
Long-Cooldown Special Attack Has Finished Cooldown
Check Long-Cooldown Special Attack Range Against Distance From Clicked Target
Long-Cooldown Special Attack On Target X, Y
Medium-Cooldown Special Attack Has Finished Cooldown
Check Medium-Cooldown Special Attack Range Against Distance From Clicked Target
Medium-Cooldown Special Attack On Target X, Y
Short-Cooldown Special Attack Has Finished Cooldown
Check Short-Cooldown Special Attack Range Against Distance From Clicked Target
Short-Cooldown Special Attack On Target X, Y

Wait For Next Key Detection Frame: Assume that there's a small delay until the game would take in keyboard input so users can't use bots to spam control input.

Detect Attack Key (Mouse Click): A mouse click denotes an attack

Has Right Mouse Key: If one of the mouse keys being pressed is the right mouse key, then right mouse key logic takes precedence over left mouse key logic

Check Current Normal Attack Range Against Distance From Clicked Target: If the player chooses a normal attack, check to see that the selected target is within range of the normal attack. The range of the normal attack can vary depending on the weapon equipped (melee or long range). (Side Note: The mouse scroll wheel selects the normal weapon currently being equipped, and there is no delay when it comes to switching between weapons.)

Current Normal Attack On Target X, Y: Attack the X, Y coordinates of the selected target

Spacebar Is Pressed Down: Check to see if spacebar is pressed down when right click happens. This changes the special attack mode to long-cooldown special attack and takes precedence over other special attacks.

Left Shift Key Is Pressed Down: Check to see if the left shift key is pressed down when right click happens. This changes the special attack mode to medium-cooldown special attack IFF space bar is not pressed down and takes precedence over the short-cooldown special attack.

Long/Medium/Short-Cooldown Special Attack Has Finished Cooldown: Make sure the selected special attack type has finished cooldown or you can't use it

Check Long/Medium/Short-Cooldown Special Attack Range Against Distance From Clicked Target: Check to see target X, Y is not out of range before attacking with selected special attack

Long/Medium/Short-Cooldown Special Attack On Target X, Y: Attack target X, Y with selected special attack

Pseudocode: Champion Attack States

Author: Sanders Lee

```
function to detect mouse click
  if right click detected
    if space bar is pressed down
      if long-cooldown special attack has finished cooldown
        if clicked coordinate is within long-cooldown special attack range
          do long-cooldown special attack on target X, Y
    else if shift key is pressed down
      if medium-cooldown special attack has finished cooldown
        if clicked coordinate is within medium-cooldown special attack range
          do medium-cooldown special attack on target X, Y
    else if neither key was pressed down
      if short-cooldown special attack has finished cooldown
        if clicked coordinate is within short-cooldown special attack range
          do short-cooldown special attack on target X, Y
  else if left click detected
    if clicked coordinate is within current normal attack range
      do current normal attack on target X, Y
```

Module: Gatekeeper/NPC

Author: Julian Brandrick

Overview

This document deals with the technical aspects of all living, non-playable characters. This currently includes all bosses, mini-bosses and minion characters as there are currently no plans for peaceful NPCs.

The details shown below are subject to change and are largely based off of logical interpretation. As of the creation of this document the design of Project Spectre is not yet complete.

Task Breakdown

- Enemy class
  - Detection radius
  - Artificial Intelligence
- Boss
  - Possible effect on surrounding minions
  - Ability structure
- Mini-Boss
  - Possible effect on surrounding minions or from nearby Boss
- Minion
  - Possible effect from nearby higher ranked monster

Attributes

Inherited Attributes

This section lists the attributes the Enemy branch would inherit from each higher level of the Base Graphical Object hierarchy. This section is not meant to presume on how the hierarchy would be arranged, only to make logical assumptions.

- Location on map
- Damage
- Health
- Speed
- Weapon
- Projectile stats
- Attack speed
- Attack range
- Status affect
- Name

Enemy Attributes

This section lists the attributes that will be contained within the enemy branch of the game hierarchy. This is not meant to be a comprehensive list and will likely be added to in the future.

- Enemy
  - Artificial Intelligence
  - Vision range
  - Guard range
- Boss
  - Ability

Currently Mini-bosses and Minions do not have any attributes that a Boss doesn’t have.

Data Structures

A data structure would be needed to contain all attributes and behaviours of an Ability. This would include a damage, target, cast range, cool-down timer and area of effect size. A passive/active toggle could also be included though this could easily be worked around with the target, and cool-down timer attributes.

Base Enemy Archetypes

Disregarding specific enemy classes, the game will contain three generic enemy base types. The boss, mini-boss and minion. These names are not final, but they adequately describe the roles of each of the enemies.

Boss

Bosses will have more HP and damage than any other enemy type and will be the most unique in design. Each boss will have a unique ability with a cool-down timer, an effect and a cast radius. They will also most likely have an effect on surrounding minions and mini-bosses, if any are close enough to be affected.

Mini-Boss

Mini-bosses will most likely be more powerful minions, but possibly could be as unique as bosses in certain respects. They will have more HP and damage than minions but, less than bosses and will not have a unique ability. Though they may have an effect on surrounding minions depending on the class of mini-boss.

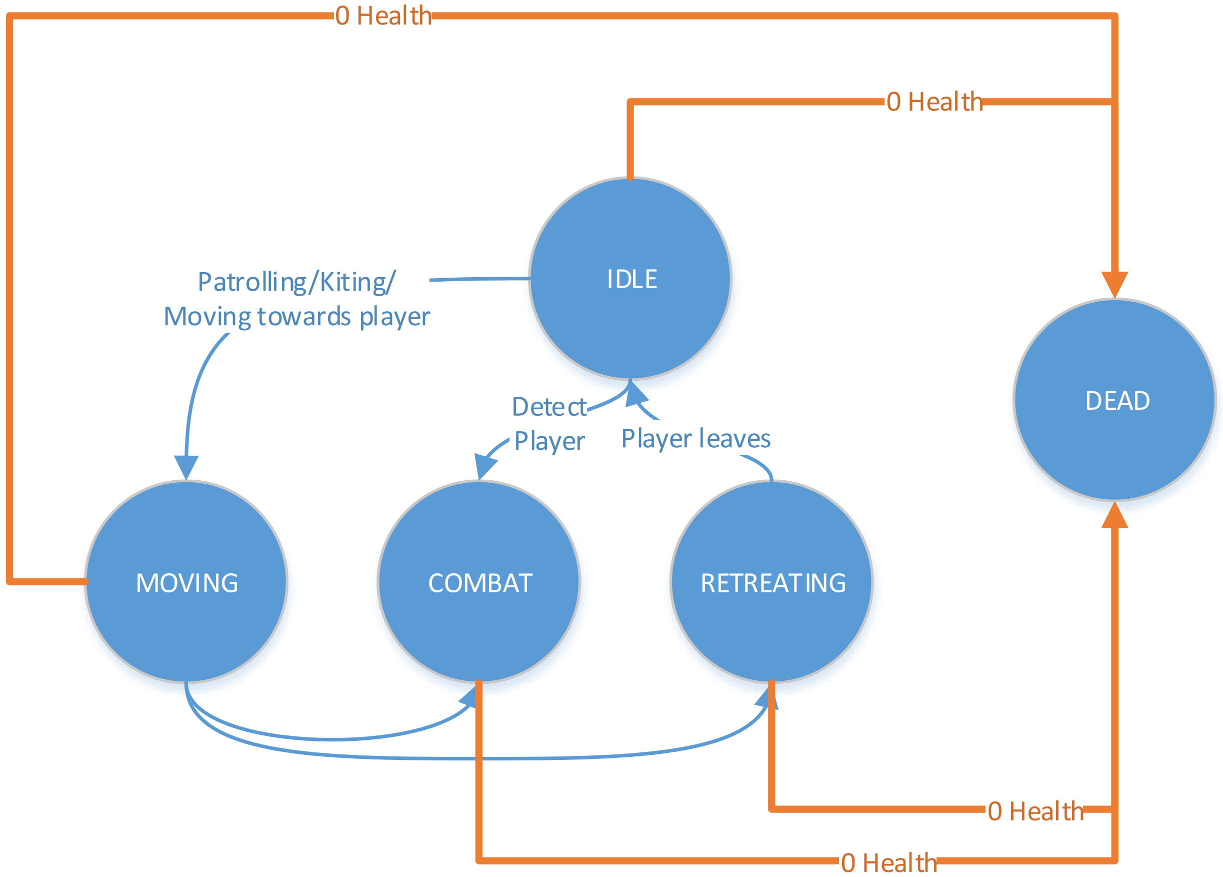
Minions

Minions are the weakest and most common type of enemy in the game. They will have the least HP and the least damage. Depending on the specific classes of minion created there could be some that interact with each other in a unique way or have unique attributes.

Gatekeeper/NPC Pseudo-Code & Design

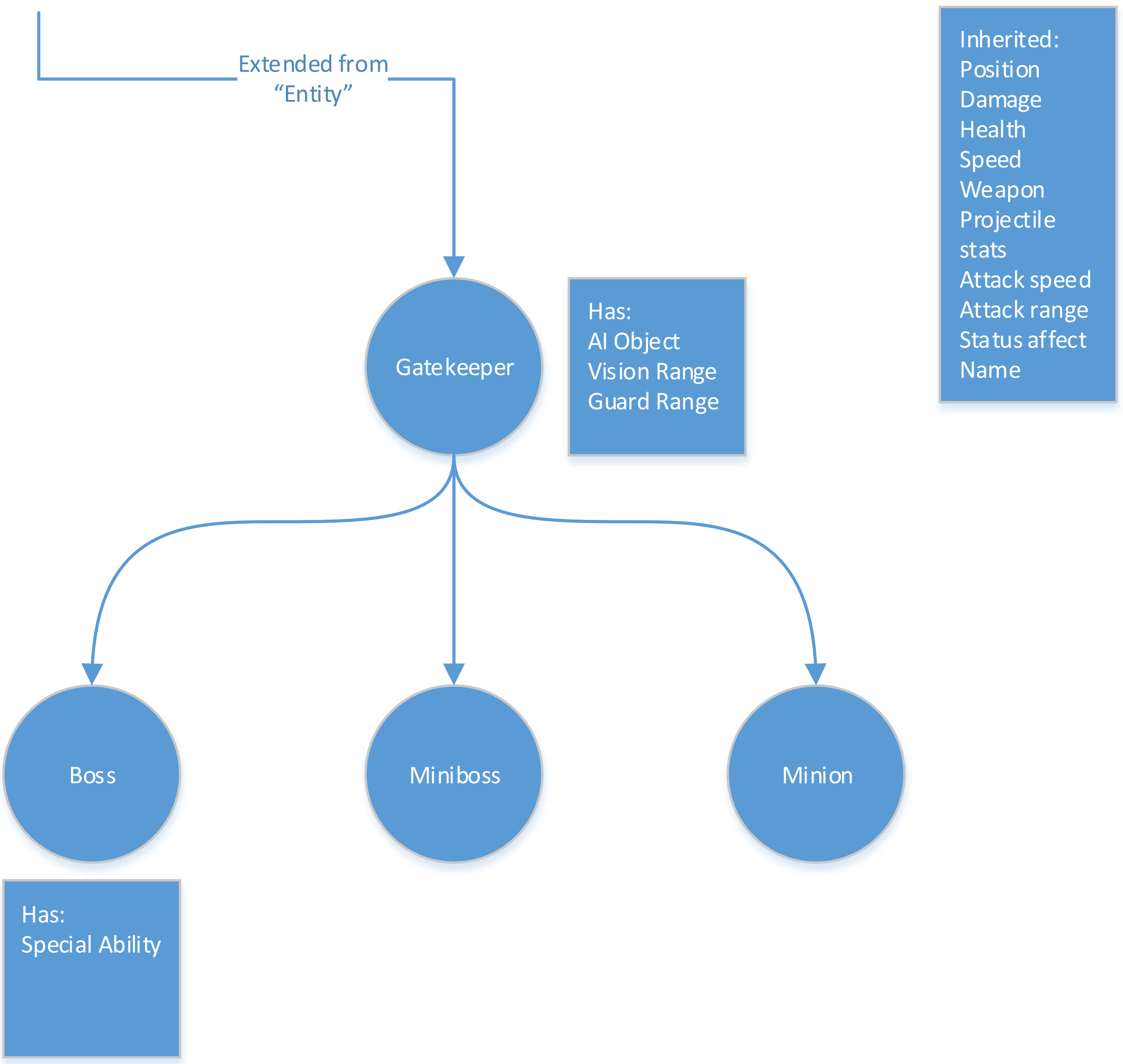
General FSM of Gatekeeper states.

Author: Filip Gutica



Gatekeeper/NPC Inheritance Hierarchy

Author: Julian Brandrick
Modified By: Filip Gutica



Pseudo-Code

Author: Filip Gutica

Class Gatekeeper

```
AI_Object
    Structure GKstruct
  Guard_Range
  Behaviour (Guard dog, Patroller etc…)
  inCombat Boolean
  Target Players
  patrol_nodes
      cooldown timer
```

Gatekeeper(…)

```
  Initialize AI object, set type, set ranges,etc
```

Function updateNPC

```
  Check type, if patroller then call patrol() function

  Call detecPlayers()
  if inAttackRange()
      enterCombat
            else
                    leaveCombat
  update the gatekeeper's position
```

Function detectPlayers

```
  Use AI_Object's scan function to detect players. Pass the guard range
  if player is detected in the guard range
              Set the target player
      Begin moving towards  targeted player
  else
      Move back to/continue default guard position or patrol
      Set the target player to null
```

Function enterCombat()

```
          Set inCombat to true
          Use the AI to attack target player
```

Function leaveCombat

```
  Set inCombat to false
      set AI to stop attacking
```

Function patrol() //Only used by patrollers

```
  Call the AI's A* function to patrol. Pass the patrol radius
```

Function inCombatRange(Attack_range)

```
Compare target players position against the attack range

Return true if player position is in the combat range
else return false
```

Class Boss Inherits from Gatekeeper

```
Special Ability
//Assuming special ability is its own object, should have attributes such as range/damage etc..
```

Boss()

```
Set stats, set ranges to boss appropriate ones
```

Function specialAbility()

```
Check the special ability range against the target player's positon
if in range
    perform special ability
    begin cooldown
```

No need to override any functions

Class Miniboss inherits from Gatekeeper

Miniboss()

```
Set stats, set ranges to miniboss appropriate ones
```

No need to override any functions

Class Minion Inherits from Gatekeeper

Minion()

```
Set stats, set ranges etc to Minion appropriate ones.
```

No need to override any functions