**DESIGN DOCUMENT**

**Zombies Ate My Students**

**Created By:**

**Alexander Davidovsky**

**Claude Gagné**

**Jonathan Lacoste**

Table of Contents

[Summary 5](#_Toc322096784)

[Overview 5](#_Toc322096785)

[Main Game Aspects 5](#_Toc322096786)

[The Role of Artificial Intelligence 6](#_Toc322096787)

[Related Games and Influences 6](#_Toc322096788)

[Characters 7](#_Toc322096789)

[The Hero 7](#_Toc322096790)

[Ms. Fields 7](#_Toc322096791)

[Zombie Teachers and Students 7](#_Toc322096792)

[Rescue individuals 8](#_Toc322096793)

[User Interface 8](#_Toc322096794)

[In Game HUD 8](#_Toc322096795)

[Technology Plan 8](#_Toc322096796)

[Software Requirements 8](#_Toc322096797)

[Hardware Requirements 9](#_Toc322096798)

[Development Software Used 9](#_Toc322096799)

[Software Architecture 9](#_Toc322096800)

[Entities 10](#_Toc322096801)

[Entity 11](#_Toc322096802)

[Hero 11](#_Toc322096803)

[Zombie 11](#_Toc322096804)

[Item 11](#_Toc322096805)

[Weapon 11](#_Toc322096806)

[Powerup 12](#_Toc322096807)

[Observer pattern 12](#_Toc322096808)

[AI 12](#_Toc322096809)

[Path finding 13](#_Toc322096810)

[Collisions 13](#_Toc322096811)

[Space Partition 13](#_Toc322096812)

[InputOutput 13](#_Toc322096813)

[Particles 13](#_Toc322096814)

[Controls 14](#_Toc322096815)

[Controls Mapping 14](#_Toc322096816)

[Inputs To On-Screen Action 14](#_Toc322096817)

[Transitive Animation States and Accepted Inputs 15](#_Toc322096818)

[Level Design 15](#_Toc322096819)

[Mechanics Analysis 16](#_Toc322096820)

[Player Mechanics and States 16](#_Toc322096821)

[Stats 16](#_Toc322096822)

[Movement 16](#_Toc322096823)

[Animations 17](#_Toc322096824)

[Strategy 18](#_Toc322096825)

[Weapons 19](#_Toc322096826)

[Items 19](#_Toc322096827)

[Power Ups 20](#_Toc322096828)

[Challenges and Puzzles 20](#_Toc322096829)

[Combat and Survival 21](#_Toc322096830)

[Zombies 21](#_Toc322096831)

[Rescues 22](#_Toc322096832)

[Artificial Intelligence 22](#_Toc322096833)

[Individual Movements 22](#_Toc322096834)

[Path Finding 23](#_Toc322096835)

[Decision Making and States 23](#_Toc322096836)

[Main States and Decisions 25](#_Toc322096837)

[Artificial Intelligence Strategy 26](#_Toc322096838)

[Animations 27](#_Toc322096839)

[Sound Mechanic 28](#_Toc322096840)

[Physics 28](#_Toc322096841)

[Player to NPCs 28](#_Toc322096842)

[NPCs to NPCs 28](#_Toc322096843)

[Player Attacks 28](#_Toc322096844)

[Environment 29](#_Toc322096845)

[Items and Event Points 29](#_Toc322096846)

[Fire Hazards 29](#_Toc322096847)

[Post Mortem 30](#_Toc322096848)

[What Went Right 30](#_Toc322096849)

[What Went Wrong 30](#_Toc322096850)

[Screenshots 32](#_Toc322096851)

[References 34](#_Toc322096852)

[Code references 34](#_Toc322096853)

[Model references 34](#_Toc322096854)

Summary

Following the neutralization of the nodding disease outbreak in North Africa, the fear of a zombie-child infestation was now gone. In the months that followed, a mutation of the parasite was discovered by the scientists involved where infected individuals developed an appetite for flesh and an instinct to pass on the parasite by eating their victims. Thought to only affect children, this mutated strain was becoming a serious threat and was found to spread quickly. A preventive drug was developed but it could not help save those who had already gone mad.

You are an elementary school teacher. Back from a long vacation, your first day back to work will take a turn for the worse. Many students and colleagues, bearers of the parasite, suddenly went mad and on a violent rampage on that day. You are now fighting for your life, trying to make it out alive to seek the drug that will save you from becoming one of them.

Zombies Ate My Students puts the player in the role of the only elementary school teacher not yet affected by the parasite attempting to escape from the infestation of zombie children and teachers. The player has to make his way through classrooms, corridors, the cafeteria and the gym collecting items ranging from a fire extinguisher, to douse the flames blocking his path, to a teacher’s personal handgun to shoot the zombies standing between him and safety and freedom.

It is a survival-action 3D game with a top down view borrowed from the play style of Zombies Ate My Neighbours and the horror survival aspect of Resident Evil. The game relies on the ability of the player to avoid zombie attacks, find the items necessary to defend himself and solve difficulties to progress through the areas of the school. The player must pay particular attention to the strategies employed by the zombies in order to ward off their attempt to trap him. Challenges of different types will be offered to the player in several places in the level to provide diversity and avoid monotony.

Overview

## Main Game Aspects

The core of the game revolves around escaping the school by finding the only safe exit in the school. Each room in the school may contain items that require a certain challenge to acquire: either fighting off zombies or solving a simple puzzle. The three main aspects of the game are exploration, combat/survival and item collection. All the NPCs in the game rely on artificial intelligence to provide interactivity and give the player the feeling that they act together against him and react to his actions. The NPCs have been made to follow certain actions in given situations while giving them the freedom to act differently if they wish. This provides some unpredictability in their behaviour that will keep the player on his toes and not expect the same thing every time.

The school is set up like a simple maze that the player must explore to progress in the game. The school map is laid out in a way that requires the player to explore it in a non-linear fashion. Obstacles will be all over the school to prevent the player from progressing and forcing him to figure out a way to get past them. The combat aspect is divided into two different kinds. Initially, the player will not be equipped to fight back against the zombies, and avoiding them will be adamant to his survival. Upon acquiring items that allow the player to defend himself, the player will need a mix of strategy and firepower in order to overcome the zombie swarms coming his way. Multiple items scattered in the school must be found by the player to progress throughout the game. Some items provide the player with the ability to solve certain puzzles while others are effective to use against zombies. Different powerups will also be available to aid in the player’s survival. The player must also find other survivors in the school and save them from a horrible death.

## The Role of Artificial Intelligence

The artificial intelligence is a big part of the system and is revolves mostly around the zombies’ interaction with the player during combat. Different types of zombies are given a different set of behaviours affecting their actions against the player. Surprisingly, the zombies have the ability think for themselves but also, as a group. The zombies make use of a decision tree based on fuzzy logic to determine their actions. Also, each decision is biased with a certain degree of randomness to avoid predictable behaviour. They will be able to pursue the player alone or in groups, use different attacks and try to gang up on the player from all sides. While the zombies are free to move wherever they feel like, they implement path finding when necessary to go where they need to go efficiently. Zombies are sensitive to sound. It is in the player’s best interest to avoid making noise in order to keep the attention of the zombies off of him.

## Related Games and Influences

Certain games have influenced the design of Zombies Ate My Students. The most obvious would be Zombies Ate My Neighbours, in which the name of the game is derived from. Zombies Ate My Neighbours is an action game For Super Nintendo Entertainment System and Sega Genesis in which a duo of teens would walk around in monster infested levels, collecting all the keys and saving all their neighbours scattered around the level before heading for the level exit, using a vast array of different weapons along the way to fight off the monsters and find secret passages. The top down view, pace of the action and NPC rescue are the main points taken from this game.

The famous Resident Evil series of games has been an inspiration for the survival aspect and the more serious and macabre tone of the story and game presentation overall. The Resident Evil series is known for being of the survival horror genre, putting Special Forces police agents in precarious and dangerously scary situations in which they must survive through zombie infestations caused by the Umbrella Corporation.

Another major inspiration for Zombies Ate My Students is the Zelda series. Each Zelda game stars a teenage boy named Link on a quest to save the princess Zelda and save the world of Hyrule from the evil clutches of Ganon. Zelda is an adventure action game in which the player must visit various temples and dungeons, collecting important items and powerups and fighting various enemies and bosses to advance to the next challenge. While the school is built on a single level, the dungeon progression and item collection is reminiscent of the Zelda games.

Characters

## The Hero

The hero is a regular 4th grade teacher of an elementary school. His name is unknown. He is an everyday normal guy without super powers who just came back from a long vacation from Europe. He has been trying to date Ms. Fields in 6th grade for the longest time and he is known for making a fantastic spaghetti sauce. A pacifist at heart, he doesn’t believe in violence but has a deadly accuracy with firearms because of his interest shooting ranges and the loud noise guns make.

## Ms. Fields

A 6th grade teacher with a passion for History, she is beloved by her students and well appreciated by her colleagues. She is the love interest of the hero. Luckily for him, she is away substituting in Canada where the parasite has not yet been reported.

## Zombie Teachers and Students

Many of the hero’s colleagues and students share a sad fate. Affected by the parasite, the high concentration of parasitic activity in many individuals at school triggered a sudden outbreak of madness and mindless violence that turned the school into a deadly labyrinth. The adults are more organized together and will attempt to trap any non-infected victims through any means necessary. Some adults will also spew out an acidic substance to incapacitate their victims from a distance. Child zombies are more rogue and run around everywhere without much purpose. However, they will gladly target the legs of the victims, holding and feasting onto them inflicting excruciating pain and slow down their victims.

## Rescue individuals

Some colleagues and students who have yet to become affected by the parasite have found refuge in certain areas of the school but their infected counterparts are on their trace. The hero must rescue them. Perhaps they will be grateful and help the hero back in return for saving their lives.

User Interface

## In Game HUD



Figure : Heads-up display design mock-up

Technology Plan

## Software Requirements

The compiled version of the game requires the Microsoft XNA Framework Redistributable 4.0 package to run on client machines. The project requires the Visual Studio 2010 development suite and the Microsoft XNA Game Studio 4.0 development package.

## Hardware Requirements

The game runs on any recent Intel Celeron, Pentium, Core2, Core i3, i5, i7 and all equivalent AMD processors with at least 2.0 GHz. At least 1Gb of system RAM is required but 2-4Gb is recommended. 2-3 years old and more recent graphics cards from ATI or nVidia should run the work fine.

## Development Software Used

* Visual Studio 2010 Ultimate 64Bit Edition
* Microsoft XNA Game Studio 4.0
* Autodesk Maya 2012
* Blender
* Adobe Photoshop CS5
* Microsoft Cross-Platform Audio Creation Tool (XACT)
* Wizdom Music’s SampleWiz (Sound Sampler)
* Subversion SVN repository hosted by Google Code

Software Architecture

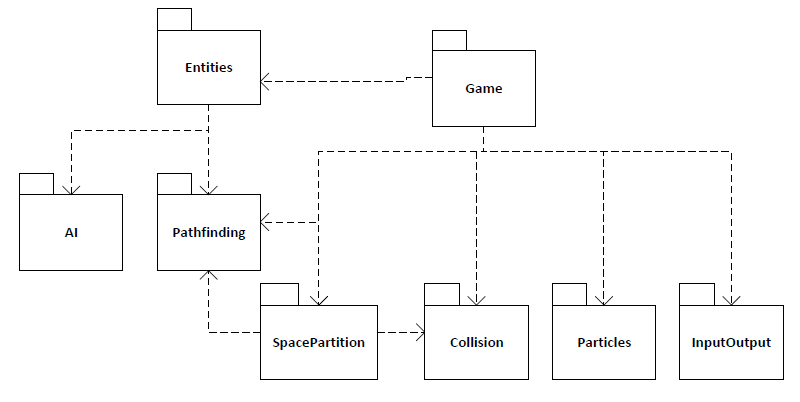


Figure : Package diagram

Figure 2 above shows the high-level package diagram of the game’s architecture. The Game package contains the Game1.cs class, which is the central point of the game, as it controls the use of all other packages.

# Entities

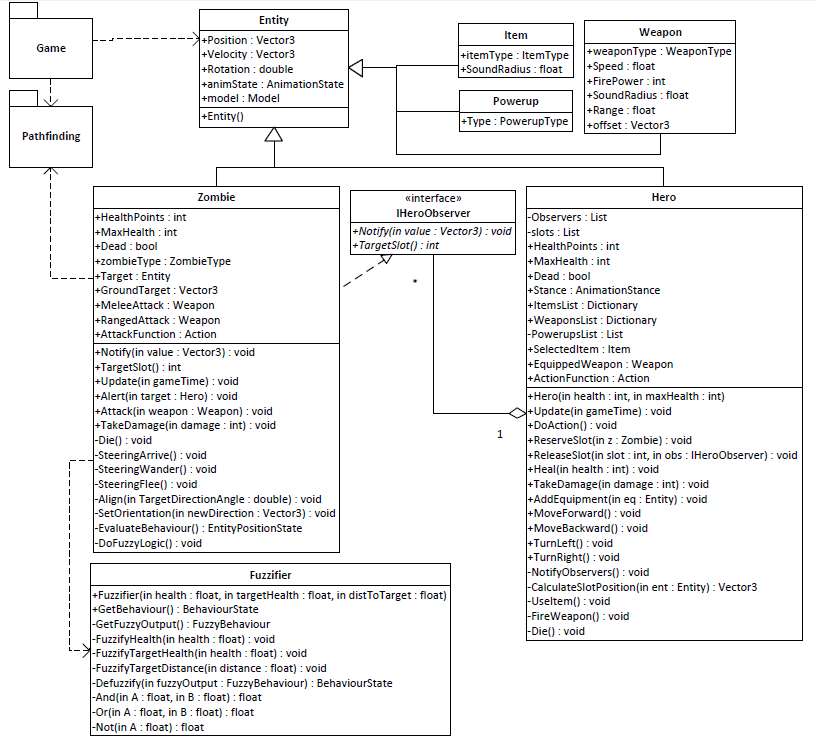


Figure : Class diagram of the Entities package

Entities are all the animated objects in the game, namely, the Hero, Zombies and items. Figure 3 shows the detailed class diagram of this package.

## Entity

This class is the parent class for all animated objects in the game. It contains the important attributes for Position, Rotation, Velocity, Model and Animation States enumeration. The Game class handles the creation, updating and drawing of all Entities.

## Hero

This is the player’s character class. It is derived from the Entity class. The Hero class contains Health Points, Max Health Points and an Animation Stance enumeration that dictates whether the player is in standing or shooting stance. It also contains several methods for movement, and managing weapons and items.

## Zombie

This is the enemies/AI character class. It is derived from the Entity class. The Zombie class contains Health Points, Max Health Points and a Zombie Type enumeration that identifies why zombie enemy the instance is. It also contains all the kinetic and steering algorithms for movements and orientation and the attributes required to drive them accordingly. It uses the path finding package to find a suitable path to reach its target. It also uses the AI package to determine which behaviour it should follow. Each possible behaviour entails a different combination of movement behaviours.

## Item

This is the item class derived from entity. While the item class does not have a need for movement, it can be animated and positioned. It contains an Item Type enumeration that defines what kind of item the instance is. Using a certain items can make noise which alerts nearby zombies. For this reason, this class contains a sound range attribute which indicates how far the sound will reach.

## Weapon

This class is also derived from Entity. This class contains the Fire Power, Shooting Speed and type of weapon. It allows the player to know how much damage it can do to a zombie and how fast he can shoot. It additionally has a sound range attribute indicating how far away a zombie can be while still hearing the player use this weapon.

## Powerup

This class, derived from Entity, simply contains a Type attribute which indicates which powerup an instance of the class represents. As powerups provide a passive bonus, a player only needs to know if they possess a particular powerup or not in order to reap its benefits.

## Observer pattern

The observer pattern is used to implement the flanking behaviour which is explained further in the Artificial Intelligence section of this document. The zombie class inherits from IHeroObserver so that it may easily keep track of the Hero’s position, rather than constantly check if the Hero has moved or not. Some implementation details are explained below.

* In a regular implementation of the Observer pattern, the Hero class would extend an abstract Observable class, which would contain the list of observers and the implementations for subscribe(), unsubscribe() and NotifyObservers(). However it is more beneficial to instead have all of the game's characters extend one class (Entity), which prevents us from being able to inherit from an Observable class.
* The Observers list is complemented by the slots list, which contains the physical locations of each of the slots around the player. This makes it easier to notify each observer of the new location of their slot whenever it changes.
* The subscribe() and unsubscribe() methods are private. They are called by the public methods ReserveSlot() and ReleaseSlot() respectively.
* ReserveSlot() checks which slot to assign to an observer, taking into account that a new slot should be a certain distance from assigned ones, and return the slot position, or -1 if all slots are taken. Most of this calculation is done in the CalculateSlotPosition() method. If a valid slot is returned, it is stored in the Zombie’s TargetSlot attribute.
* ReleaseSlot() is simply a facade over unsubscribe(), and is called either when the Zombie is killed or changes from a pursuing behaviour to another. By passing the TargetSlot attribute into the method, it enables the Hero class to know which IHeroObserver to remove and which slot to free.
* Whenever the Hero’s position is changed, it calls notifyObservers() as per the usual observer pattern behaviour. notifyObservers() will then, for each IHeroObserver, use the slots list to find the new Vector3 associated with the observer’s slot, and send this value to the Zombie via its notify() method.

## AI

The AI package consists only of the Fuzzifier class, which is also shown in figure 3. This class implements fuzzy logic by taking as input several factors of the Zombie’s environment and producing an output in the form of the BehaviourState enumerated type. The fuzzy logic used is described in detail in the Artificial Intelligence section of this document.

## Path finding

The Pathfinding package contains classes used implement the A\* algorithm. It uses the Node class to translate a world position into a traversable node. The NodeConnection class is used as well to represent the movement between two Nodes, and the cost of doing so.

## Collisions

The Collisions package contains the Primitive class which represents an object in the world space with strict boundaries. The Sphere and Box classes inherit from Primitive. Zombies and the Hero are represented by Spheres, and all static objects are implemented as Boxes. This package provides methods for detecting collisions between any Primitive class, which return an instance of the Contact class. A Contact indicates the position of contact between two Primitives, as well as the contact normal and interpenetration depth.

## Space Partition

The SpacePartition class is used to implement a Quad Tree to divide the game world into smaller sections. This allows the game to quickly retrieve nearby Primitive objects from a given position, or also nearby pathfinding Nodes.

## InputOutput

This package combines many classes used to display and retrieve information from the player. It handles the Camera class, the HUD class (heads-up display) and the Sounds class to output information to the player, as well as the KeyboardInput class to validate any input.

## Particles

This package is used by the Game for displaying particle effects, which are used for fires and the fire extinguisher. The use of this class is described further in the Particle System section of this document.

Controls

## Controls Mapping

#### Standing Stance

Up Arrow: Move Forward

Down Arrow: Move Backward

Left Arrow: Turn Left

Right Arrow: Turn Right

Space: Use Selected Item

#### Shooting Stance

Up Arrow: Nothing

Down Arrow: Nothing

Left Arrow: Rotate Aim Left

Right Arrow: Rotate Aim Right

Space: Fire Active Weapon

#### General

Tab: Select Next Item

W: Select Next Weapon

Left Shift(Hold): Change to shooting stance

Left Shift(Release): Change to standing stance

## Inputs To On-Screen Action

All the keyboard inputs entered by the player are managed by the XNA KeyboardState object. The player’s inputs are filtered based on the character’s actions stance and animation states. At the top level, the input management system first rejects all inputs not permitted by transitive animation states. If the character is not in a transitive state or if transitive inputs are allowed, the system goes through a stance filter. Those inputs will be interpreted and trigger the action mapped to the appropriate stance enumerated previously.

### Transitive Animation States and Accepted Inputs

* Walking (Standing)
  + Arrow Keys, Tab, W
* Shooting
  + Tab
* Using Item (Standing)
  + No Inputs
* Damaged
  + Tab
* Shaking off child zombie
  + Arrow Keys, Tab
* Dying/Dead
  + No inputs

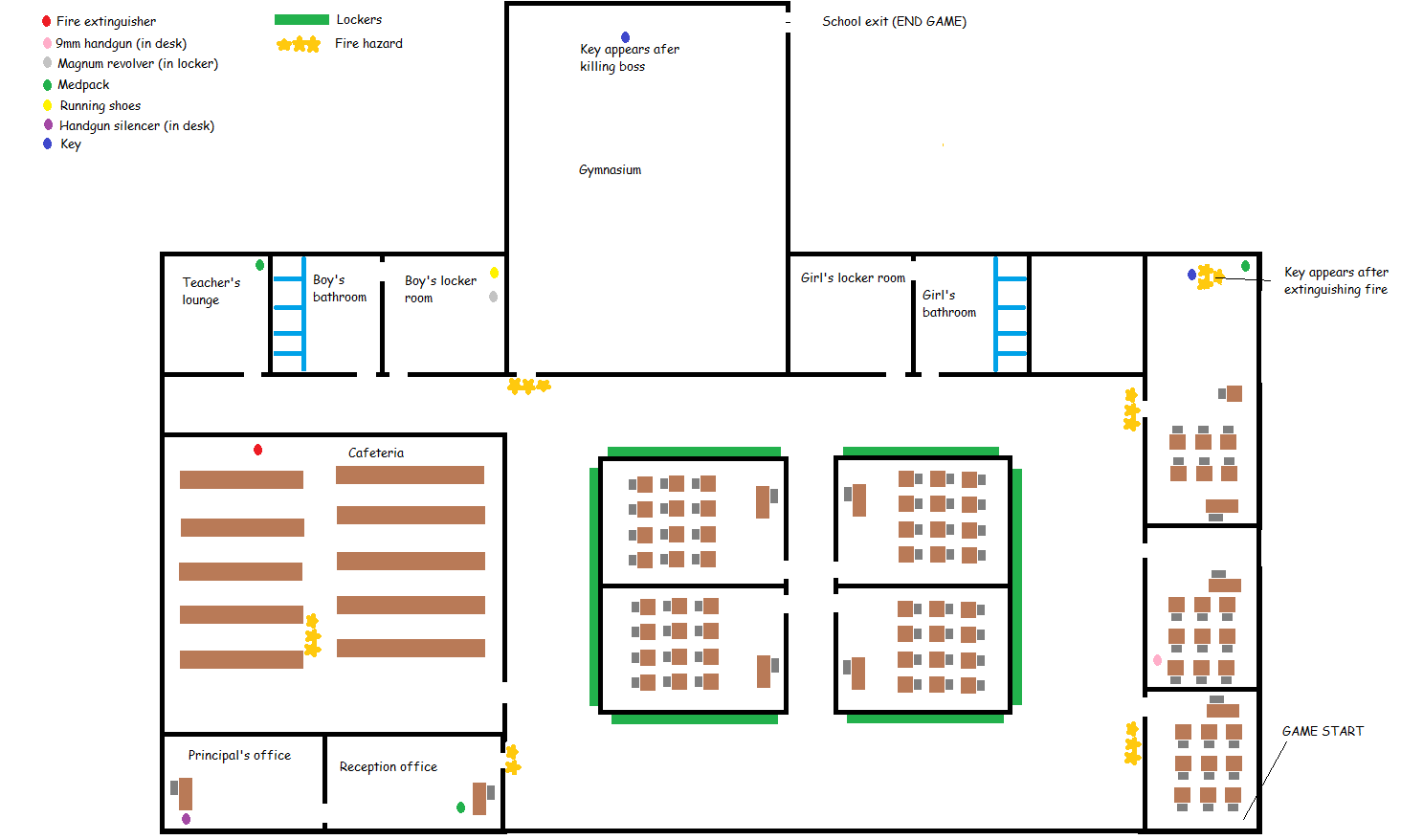
Level Design

Figure : Level Map

The conceptual map of the game’s level is shown above in figure 4. The intended progression of gameplay is as follows:

The player begins in the classroom in the south-east corner of the map. As they begin unarmed, they should acquire the 9mm handgun in the room adjacent to the starting classroom. Next, they will require the fire extinguisher to access the rooms blocked off by fire. They will have to go to the cafeteria for this, at the west side of the school. Once they have the extinguisher, they will need the key located in the north-easternmost classroom, where they need to extinguish the fire covering the key.

At this point the player is free to extinguish the fire blocking the gymnasium door and exit the school. There are however additional powerups they may obtain in the administration section of the school in the south-west corner, as well as the boy’s locker room in the north-west. These powerups are recommended to use before facing the zombie horde in the gymnasium.

Mechanics Analysis

## Player Mechanics and States

### Stats

#### Life

This is the character’s life gauge. It is set as a value of 1000. Zombie attacks and hazards will lower this value. When it reaches 0, the hero dies. It can be replenished with medical kits. The bar is represented on the top left corner of the screen. A letter grade is associated with his current health level from D to A, where A represents full health. When the bar is nearly empty, the letter F will appear signalling the character’s critical condition.

#### Speed

This is the walking speed value for the character. This attribute is also used when repelling kid zombies stuck on the hero. Sneakers will enhance both the walking speed and repelling ability.

### Movement

The player can control the character’s movement using the control scheme outlined in the Controls section. The movement is derived from the Resident Evil style where the player uses the left and right arrows to rotate and the up and down arrows to move forward and backwards. It compromises on flexibility of movement and speed for more control often required in those types of games.

### Animations

Whenever the player executes an action, an animation occurs. During the animation sequence, it is not possible for the player to do certain actions depending on which animation is current undergoing. The player starts with the Idle state to indicate that no animation is happening.

#### Walking

This is the normal walking animation for the character.

Footsteps sounds should be heard. The initial shoes make a “tack tack” type sound due to their wood soles. The Sneakers make much softer sounds.

#### Standing to Shooting Stance

This animation goes from a standing to a position where the character is holding the selected weapon in his hands and his legs set to retain posture after firing. The last frame of animation remains the position in which the character will remain as long as he is in this stance. The player cannot move or use items while in this stance. This animation should be executing once when pressing and holding the shift key.

#### Shooting to Standing Stance

This animation goes from a shooting position to a standing position. This is the reversed animation from Standing to Shooting Stance. The last frame of animation remains the position in which the character will remain as long as he is in this stance. The Player cannot use a weapon in this stance but can walk and use items. This animation is executed only once when releasing the shift key

#### Shooting (Guns)

This animation is occurs when the player is in the shooting stance and attempts to shoot. The character’s arm jerks backwards very quickly, mimicking the animations of a person’s movement under the influence of gun recoil. This animation is only for guns. The Fire Extinguisher doesn’t trigger this animation when using it. The recoil of the 9mm gun creates a quicker animation with less arm movement. The Magnum creates a long animation with much more exaggerated arm movement due to the much stronger recoil impulse.

The weapons sound should be heard when shooting. It is activated once per shot for the guns and is continuous for the extinguisher until done.

#### Damaged

At any stance or animation state, the character will enter the damaged animation whenever he suffers damage. The animation consists of simply arching backwards slightly, head going backwards and arms going in the air. Being Damaged resets the stance to standing.

The character’s pain sound effect should be played. This animation should be quick and damage from other enemies is nullified while the player isn’t back to his a normal stance.

#### Dying

If the character suffers too much damage he dies and falls on the floor.

The character should scream a death cry.

### Strategy

The player must keep track of his life bar to ensure his survival by avoiding enemies and environmental hazards. He should use medical kits when his life is too low.

Everything the hero does will generate some levels of noise. The closer the playing is to enemies to more likely they are to hear him walk by and attack him. Noise is also triggered when bumping into walls or obstacles with a varying degree of loudness. The louder the noise is the farther its reach for zombies to hear. Firing weapons also creates very loud noises. Sneakers and the Silencer help reduce the noise created when walking and shooting to provide an advantage to the player. Based on certain situations, the player will need to make strategic choices as to what to do based on the loudness level of the actions to be executed and the risk they entail.

The 9mm gun while weaker than the magnum proves to be very useful as a long range weapon when the silencer is equipped. The magnum should be used at closer range for maximum efficiency.

With these tricks in mind, the player should have everything required to complete the game without dying too much.

## Weapons

#### Bare Hands

The hero initial starts his adventure bare handed. It leaves him completely helpless against the zombies. Running and hiding is smarter than attempting to punch one’s way through a swarm of zombies. This isn’t really a weapon but puts emphasis on the defenselessness of the player while not equipped with a weapon.

#### 9mm Handgun

Found in the belongings of a fallen colleague, this weapon is fast with low recoil impulse and good accuracy but a somewhat weak firepower. Three or four shots of this weapon can take a zombie out. The weapon is meant to be the initial weapon found in the game. It is good for long distance shooting but initially too loud to take out many zombies easily as they will reach the player before he can kill them. A silencer updates makes it much more versatile.

#### Magnum

Found in a student locker, this weapon is slow and has a high recoil impulse and low accuracy but has fierce fire power. One well-placed shot can fell a zombie from a close range. 2-3 shots may be required from a longer range. This weapon is meant to mow down zombies in close range with ease. Combining both the Magnum and the 9mm gun are important to maximize survival.

## Items

#### Fire Extinguisher

The fire extinguisher is needed for the hero to make his way through the flames of certain areas in school. It serves as a progression item only for the moment.

#### Keys

Two keys are hidden in the school. The first key unlocks a locker in the school corridors where the Magnum gun is found. The key can be found hidden in a class room, engulfed in flames. The Second key is required to unlock to condemned exit of the gymnasium to escape to safety. The key is dropped by one of the zombies roaming in the gym.

## Power Ups

#### Silencer

The silencer is an add-on for the 9mm handgun that reduces the noise it makes when using it. It allows the hero to fire away without alarming all the zombies around.

#### Sneakers

A functional pair of shoes that provides a movement speed boost for the hero. This helps deal with speed reduction caused by kid zombies as well, and makes the player generate much less sound when walking.

#### Med Kit

Rare to find, these collectable items will restore some of character’s health. Randomly restores between 25% and 75% health.

## Challenges and Puzzles

#### Fire Blockades

Some areas of the school are blocked by fire preventing passage to the later areas of the game. The player must find the extinguisher to proceed further. This is a simple progression control feature that fits with the tone and feel of the game.

#### Zombie Waves

At certain times in the game, the player will find himself in a “choke point”, where he will be forced to defend himself, but where doing so will alert a large number of enemies to his location. He will have to fight his way through to survive.

#### Gymnasium Brawl

This is final challenge of the game. Many zombies are gathered in the school gym, the biggest room in the school. Fortunately for the hero, the Gym has the only door that will lead the hero to safety. Unfortunately, it’s so filled with zombies that it will be the fight of the hero’s life. It is also rumoured that the school’s principal, his strong authority granting him enhanced abilities, lurks in this room.

#### Noise obstacles

Scattered around the game will be obstacles that will make noise whenever the hero steps on or bumps into them. These noises will alert zombies of the hero’s presence and as such, should be avoided. It is a rare feature that is often underused and overlooked in video games.

## Combat and Survival

Zombies are found everywhere in the school and pose a serious threat to the player’s survival. The quantity of zombies found in various locations and combinations will force the player to use different tactics to get through based on the items and weapons at his disposal.

An important part of combat is the mandatory switch to shooting stance when attempting to kill zombies. The shooting stance is a quick animation that shifts the player’s standing position to one where he hold his equipped weapon, ready to shot. This switch is meant to slow down the player’s action responsiveness to add an element of strategy to fighting and to separate item use from weapon use.

Because of the need to switch stances between attacks and item usage, the player must be careful on when and where to change into the desired stance. For example, using a medical kit can take some time in the heat of battle that leaves an already vulnerable player at risk.

The player is given a limited amount of life and he must avoid getting damaged too much or the game will end. Medical kits are available through the game but their effectiveness is random and they are scarce. This is meant to create tension by providing some relief for taking heavy damage but forces the player to anticipate the actions of his enemies to avoid getting wounded as much as possible.

## Zombies

#### Adult Zombies

Adult Zombies are often found in groups. They also tend to react as a group. Adult zombies have two means of attack. They bite at close range and spew acid at long range. Their goal as a group is to surround the player, prevent him from moving away to a safe spot and deal as much damage as they can. They are quite slow and easy to outmanoeuvre but a swarm of these monsters is difficult to avoid without getting into a fight. They are attracted to sounds and are capable of finding their way to the source with relative ease. They are the most common enemy found throughout the game. They will wander aimlessly until something catches their attention. It is possible to pass by them if the player doesn’t make noise. Adult zombies are able to regenerate their wounds fairly quickly if left unattended for a while. During a fight, they may retreat to regenerate before coming back in the fray.

#### Kid Zombies

Kid Zombies are rogue and independent. They are fast and small targets. Their only attack is to run towards the player attempting to slow him down by getting in the way and biting his legs. Kids are difficult to get rid of if they find their way to the player and will slowly tap into the player’s life bar. The player is in serious trouble if he is caught by a kid while being surrounded by adults. Kids are often found running around alone and are more attuned to sounds than regular zombies. They share a scaled down version of the adult model.

#### Principal Zombie

This is the final boss of the game fought in the gym. It is surrounded by hordes of zombies. It is faster, stronger and much more resistant than other zombies. It has the same attacks as the normal adult zombies but they can kill the player very easily if he is not careful. It shares the same adult zombie model, but at a larger scale.

## Rescues

During the game, the player will encounter lucky colleagues and students who have not yet been affected by the parasite. They are often found in situation of danger and saving them is a priority as they give good incentives to do so. Some may provide some items to help along the way, others will help reduce and even eliminate zombie respawn in certain areas.

Artificial Intelligence

## Individual Movements

All zombies implement the Arrive, Pursue, Flee and Wander steering behaviours, which are all delegated to Seek. In their passive behaviour, zombies use the wander behaviour to walk around. When actively seeking the player, zombies will switch to Pursue. When near the player, they will use the Arrive behaviour to close in on the player, preparing for attacks. Adult zombies also have the flee behaviour which they use when critically injured.

## Path Finding

The enemies in the game are not bound to any graphs and are free to move however they feel like. However, they are subject to collision detections in order to keep them within the bounds of the level. Zombies rely on path finding when they need to close in on a target. A set of POV nodes stored in a Quad Tree, for performance reasons, will be placed all around the rooms of the school to serve as movement markers to move closer to their target. Zombies within a certain distance from their target will cast rays to verify clearance and move in directly towards their target if the path is clear. If obstacles are in the way, they will resume path finding until they find a position that allows them to attack their target.

## Decision Making and States

The zombies rely on sets of states that govern their immediate behaviour. Each state has its own decision tree that allows the zombies to make decisions based on their stats and information of the world around them.

The state of a zombie is decided through fuzzy logic. All stats are made in a way such that they become part of fuzzy sets. These sets are used to analyze a situation and determine the best possible action at any given time with a touch of randomness thrown in for a bit of unpredictability.

The AI logic uses three input variables to determine NPC behaviour which are the NPC's health, the target's health, and the distance from the NPC to its target. The fuzzification process uses membership functions to map each input variable to a degree of membership. The membership functions for health and distance are detailed in figures 4 and 5.

Figure : Membership function for NPC and Player health

Figure : Membership function for distance between NPC and Player

The following axioms are required to analyze the input, where A and B are between 0 and 1:

* A OR B = Max(A, B)
* A AND B = Min(A, B)
* NOT A = 1 - A

The fuzzy input is then analyzed using the following rules to assign degrees to different categories of outputs:

* Out of range AND Self Healthy = Close in
* (Self Critical AND Target Healthy) OR (Melee range AND NOT Self Healthy) = Flee
* (Melee range AND Target Healthy) OR (Projectile range AND Target critical) = Melee Fight
* (Melee range AND NOT Target Healthy) OR (Projectile range AND Target critical) = Melee Troll
* Projectile range AND Target Healthy = Ranged Fight
* Projectile range AND NOT Target Healthy = Ranged Troll

Where each output category is a set of possible behavioural states which are decided randomly:

* Close in: MeleePursue, MeleeCreep, Wander
* Flee: Flee, RangedPursue
* Melee fight: MeleePursue
* Melee Troll: MeleePursue, RangedPursue, RangedCreep, Flee
* Ranged fight: RangedPursue, MeleePursue
* Ranged troll: RangedPursue, MeleePursue, MeleeCreep, Flee, Wander

## Main States and Decisions

#### Wandering

Zombies are set to this stage initially in most situations and will simply use the Wander steering behaviour while that state.

At regular intervals, they let a moaning sound out.

#### Pursuing

Zombies made aware of the player’s presence or in proximity to a rescue character will enter pursue mode. In this state, the zombies will first make use of the pursue behaviour when further away.

At all time during this state, they will cast rays for detecting collisions with other zombies, obstacles and the player. They rely on path finding in this state until they are close enough to their target and have clear passage to it. When closer they use the steering Arrive behaviour until they reach their target. At destination they will switch to the Attacking State. The zombies align themselves with the path during travel. Adult Zombies have a certain radii in which they can decide to enter attacking mode from a distance to spew acid. This will depend on whether the zombie is using the “Ranged Pursue” or “Melee Pursue” behaviour.

#### Attacking

Zombies in attacking state will have a timer countdown that regulates the number of attacks they can do in a given time. When the timer is up, they will execute their attack animation based on the attacks available. Adult zombies can attack at close range or spew acid from a distance as well as in close range if they desire to do so. In this phase, the zombies attempt to align themselves with the position of the player’s character.

In this state, zombies monitor their distance from the player and will revert back to pursue if they are too far. If they take too much damage, adult zombies will switch to a fleeing state.

#### Fleeing

This state is exclusive to adult zombies. They will make use of path finding to find a spot a certain distance from their current position as far from the player as possible to regenerate

#### Dying

When zombies suffer too much damage, they are killed and will fall on the ground and remain motionless. The zombie will remain in this state until it is out of screen view. When out of screen view, the zombie is removed from the list of monsters permanently.

When dying they let moaning sound.

#### Scared

This state is exclusive to rescue characters. They remain in their position and simply shake and hide their faces in fear.

#### Thankful

This state is exclusive to rescue characters. They remain in their position and simply wave in a thankful manner.

## Artificial Intelligence Strategy

#### Flanking

Whenever a zombie is in a pursue or attacking state, it will validate its membership in a zombie group sharing these states. All zombies in an active group will coordinate their efforts against the player in an attempt to surround him. The way this is achieved is by using a radius around the player called a flanking ring with dynamically generated positions around the player called “slots”. The number of slots is aimed to be 6 and will be at equal distances from each other on the circle. The slots will only depend on the position of the player and not its orientation.

Whenever a zombie in pursue state is close enough to the player, it will verify the slots available, preferring those first in their direct line of sight. When a slot is selected, the zombie will bind to this slot and use Arrive to reach its destination and begin his attack. All other subsequent zombies in proximity will check the slots for a potential position and bind with a slot that is strategically suited to surround the player and prevent him from moving to safety.

Because the player is likely to move during this time, slot positions are updated by the pursuing zombies each frame until they change to an attacking state. In this state, as long as they are in range of attack, they will remain in position. When they are out of range, they will try to reposition themselves to the slot.

If the player moves past a certain distance from zombies bound with a slot, they will be dropped from the flanking slots. From there, the pursue state will resume as normal. Killed zombies are also dropped from their binding with a slot.

In the case where all slots are bound, Adult zombies may choose to attack from a distance. All others will keep walking in the direction of the player, bumping in other zombies in their way until a slot is free.

## Animations

#### Walking

This is the normal walking animation for the zombies.

At regular intervals, they let a more intense moaning sound out.

#### Attacking

Adult zombies attack at close range by leaning towards the player to bite them. The animation for spewing acid is the same but green particles are thrown.

Kid Zombies will simply put their arms forward to attach themselves to the player and be bound to the player’s position until they get shaken off the player’s legs.

The principal zombie attacks in the same way as adult zombies.

Upon attacking, they let will use a grunting sound.

#### Damaged

At any state or animation state, the zombie will enter the damaged animation whenever he suffers damage. The animation consists of simply arching backwards slightly, head going backwards and arms going in the air.

They will let a pain moan when attacked.

#### Dying

If the zombie suffers too much damage he dies and falls on the floor.

The zombie lets a death moan out.

## Sound Mechanic

Whenever a player executes an action or collides with an obstacle, a sound occurs and a circle of sound will be expanded away from the player in an outward motion. The size of the circle is based on how loud the noise is found to be. The radius of the circle is quickly expanded from 0 to n radius where n is a float value assigned to each noise event. At any time during the expansion of the circle, all zombies that are in wandering state will automatically turn to a pursuing state. As soon as the radius of the circle hits its maximum, the circle is dissipated. This circle is implicit and not shown visually in the game. All obstacles and actions in the game are assigned a Noise property that defines the noise to play and its loudness.

Physics

## Player to NPCs

The player and zombies collide together using custom made sphere collisions for simplicity purposes. Upon collision, the contact information is generated and an impulse is calculated always in favour of the player depending on his speed. As such, when the player bumps into a zombie, he will put it towards the direction is his going, alerting the zombie of his presence in the process if he isn’t already aware of the player’s presence.

## NPCs to NPCs

Zombies also collide together to ensure that they never overlap. As with the player, zombies collide using custom made sphere collision detection. Upon collision, zombies are always given an equal impulse to push them away from each other so they can keep a certain distance at all times.

## Player Attacks

When the player is in shooting stance, a green line is projected towards the orientation vector of the player. When the player is shooting, first, the collision with the closest zombie is resolved. If a zombie is hit, a second collision check is made to ensure that nothing is in the way between the player and the zombies. As such, it is not possible to shoot zombies through walls or through any other objects found in the level even if technically it would possible to shoot a bullet above a chair or table. This is a conscious choice that adds more difficulty to fighting in crowded areas.

## Environment

All the walls and collision objects in the level are implemented using custom made collisions boxes. Both zombies and players use spheres to collide with the environment boxes. Upon collision, a simple corrective impulsive in the direction of the contact normal is applied of the size of the penetration depth to prevent the player and NPCs to go through obstacles and also allows them to glide smoothly through along the obstacles instead of abruptly stopping them in their movement.

## Items and Event Points

Item collection is simply done by detecting the distance between the hero and the item location without a given acceptable radius. If the hero’s distance is smaller than the threshold, the item is collected. The only event point in the game is the escape spot at the exit door in the gym. This point uses a sphere for collision and the game will end if the player carries a key with him when he enters this spot.

## Fire Hazards

The fire hazards around the level are implemented using, custom made collision boxes and are always added to the environmental collision checks. Fire hazards carry a special tag to differentiate them from the rest of the collision boxes so that damage is dealt to the hero is he touches them. The fire extinguisher item also projects a white mist that suppresses the fire over a short amount of time. The collision for the mist is made using a big collision sphere projected at a distance from the orientation of the player and covering most of the radius of the mist. If the sphere collides with one of the fire hazards, the emitter’s particle group’s lifespan is reduced until it becomes 0 at which point the particle emitter for the fire is turned off and the collision box for the fire hazard is removed.

Post Mortem

## What Went Right

#### 1. Design

From the very start, we insisted on having a complete design before starting development in order to judge the scope of the project given the time remaining to complete the project. The initial work allowed the team to get a good vision of the game and to avoid confusion, backtracking and work that would only waste time. Initially intended for a team of four, the design proved to be almost a perfect match in terms of workload as very few components were dropped due to lack of time.

#### 2. Code base

Over the course of the semester, the team acquired a lot of code that was developed with reusability in mind. The team avoided reinventing the wheel by reusing classes built for path finding, collision detection, AI behaviors and heuristics, character handling, particle effects and spatial partitioning. This allowed the team to work on creating the game specific logic component almost immediately.

#### 3. Team cohesion

Based on the design, the team was able to split the work efficiently to maximize the efforts and minimize overlaps in who did what. The constant communication between everyone over the duration of the project made it easy to get a clear idea of where the project was going and how much work was getting done.

## What Went Wrong

#### 1. Time constraints

The team started on the project very late in the semester since other classes and some of the members’ full time work schedule left a lot less time than expected to work on the project. Also, because we weren’t aware of one of our teammates dropping out two weeks before the submission of the project, we had to reassign extra work to the remaining members. It was quite stressful until the very end but we knew we had something we were satisfied with despite lacking some polish in the presentation of the game.

#### 2. Modelling and Animations

Animations proved to be far more complex than expected. As none of us are experts in 3D modelling, rigging and animating, the team spent a lot more time on getting this part right than we should have. We insisted on using free for use professionally made models with embedded textures and animations. However, things were very difficult to implement due to a lack of knowledge and experience with such models and the software required to animate them such as Blender and Maya. Finding properly animated and textured models that were free and useable for the purpose of the game was extremely difficult to do. A seasoned Maya user known by one of the team members helped save a lot wasted time by providing advice and guidance just in time to allow the team to use these models inside the game. Chances are that the current animations and models would have probably not made it into the game and wasted a lot more time than it did with this needed help. The entire completion state of the project would have been affected negatively. In the end, some animations were buggy for unknown reasons and others had to be dropped due to time constraints. We consider the hero to be Mr. Fantastic in disguise because his limbs become very long and deformed when being attacked or dying.

Many models used for the game had a polygon count that was too high for game use and had to be scrapped or redone entirely with a low polygon count without sacrificing quality so that models would retain a similar feel to them. This also proved to be a long and frustrating step that was unexpected until the end of the project which had the team waste time.

#### 3. Path Finding and Rescue Victims

While the team had sufficient code to do efficient path finding, implementing that into the zombie logic conflicted with the fuzzy logic and flanking components already in place. Because this was implemented too late into the project, the team decided to drop the path finding altogether to spend more time on finishing other required features. Also, creating the nodes and the links between them proved to be a lengthy process that turned out to be done in vain.

Initially, rescue victims were planned to make into the game that would provide various bonuses and advantages to the hero if he managed to save them on time before the zombies got a hold of them and killed them. Due to lack of time but also because the idea turned out to be not as good as intended, it was decided that these rescues victims would be dropped entirely from the project.

Screenshots



Figure : The player extinguishing a fire with the extinguisher

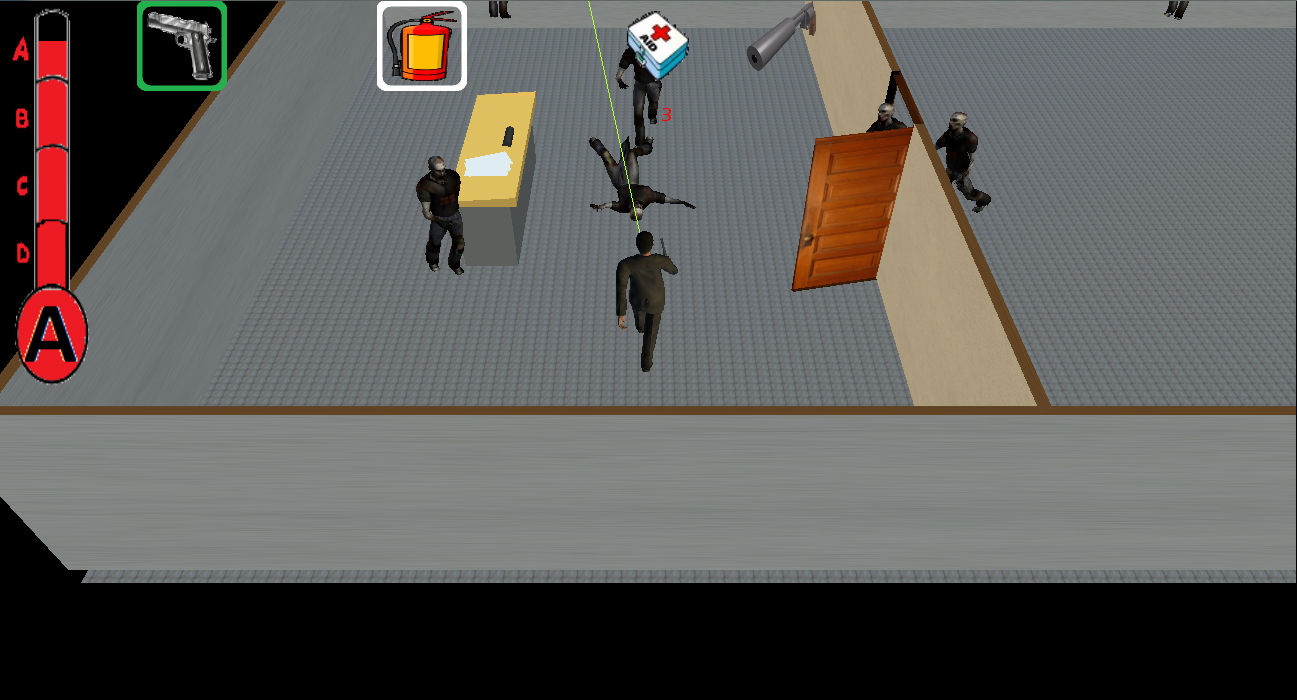


Figure : The hero fighting killing zombies with the handgun and silencer



Figure : The final battle with the zombie principal and his minions in the gymnasium before the hero can escape.

References

## Code references

* Collision detection
  + Developed by Claude Gagné COMP 476 Assignment 3
  + Modified and implemented by Alexander Davidovsky
* Entities Folder
  + Developed by Claude Gagné COMP 476 Assignment 1,2,3
  + Modified and implemented by the entire team
* Particle System
  + Developed by Claude Gagné COMP 477 Assignment 3
  + Modified and implemented by Claude Gagné
* Path Finding
  + Developed by Claude Gagné COMP 476 Assignment 2
  + Implemented by Alexander Davidovsky and Jonathan Lacoste
* Space Partitioning
  + Developed and implemented by Claude Gagné COMP 476 Assignment 3
* Fuzzy Logic
  + Developed by Alexander Davidovsky
* HUD
  + Developped by Jonathan Lacoste
* Sounds
  + Developed by Jonathan Lacoste
* Animation processing
  + Implemented by Jonathan Lacoste
  + Based on the App Hub code sample from <http://create.msdn.com/en-US/education/catalog/sample/skinned_model>

## Model references

* Conference Table Model
  + <http://www.turbosquid.com/FullPreview/Index.cfm/ID/198443>
* Metal Doors
  + <http://www.turbosquid.com/FullPreview/Index.cfm/ID/418223>
* Toilet
  + <http://www.turbosquid.com/FullPreview/Index.cfm/ID/173924>
* Locker Room Bench
  + <http://www.turbosquid.com/FullPreview/Index.cfm/ID/452223>
* Bathroom tiles
  + <http://www.turbosquid.com/FullPreview/Index.cfm/ID/325233>
* Urinals
  + <http://www.turbosquid.com/FullPreview/Index.cfm/ID/657152>
* Doors
  + <http://www.turbosquid.com/FullPreview/Index.cfm/ID/612776>
* Gym Walls
  + <http://www.turbosquid.com/FullPreview/Index.cfm/ID/515446>
* Med Kits
  + <http://www.turbosquid.com/FullPreview/Index.cfm/ID/622126>
* 9mm pistol
  + <http://www.turbosquid.com/FullPreview/Index.cfm/ID/657083>
* Sneakers
  + <http://www.turbosquid.com/FullPreview/Index.cfm/ID/295457>
* Key
  + <http://www.turbosquid.com/FullPreview/Index.cfm/ID/246564>
* Fire Extinguisher
  + <http://www.turbosquid.com/FullPreview/Index.cfm/ID/540469>
* Computer Room
  + <http://www.turbosquid.com/FullPreview/Index.cfm/ID/247284>
* Classrooms and teacher lounge TV
  + <http://www.turbosquid.com/FullPreview/Index.cfm/ID/247291>
* Gymnasium
  + <http://www.turbosquid.com/FullPreview/Index.cfm/ID/502569>
* Locker
  + <http://www.turbosquid.com/FullPreview/Index.cfm/ID/505340>
  + Almost completely redone to reduce poly count
* Magnum
  + <http://www.turbosquid.com/FullPreview/Index.cfm/ID/621822>