Design Documentation

Version 1

Fall 2012

Developers

Willie Ling

Jeff How

Jonathan Bergeron

Addison Rodomista

Table of Contents

Executive Summary 1

Overview 1

Related Games 2

Player Composites 3

World 3

Characters 4

- The Hero 4
- The Enemies 4
  - The “Hellhound” 4
  - The “Demon” 6
  - The “Fallen” / “Half Demon” 7

Progression Graph 8

Art Direction 9

- Concept Art 9
- Reference Art/Inspirations 10
- Themes 12
- Art Constraints from Technology 12

UI Storyboards 13

Tags & Dialog 13

Technology Plan 14

- Rendering 14
- Level 14
- Physics 14
- Collision Detection 15
- Animation 15
- Other tools 16

Software Architecture 17

- Core Classes 17
- Entity Classes and Subclasses 18

Weapon-Related Classes 22
UI-Related Classes 23
Other Classes 24
External Classes 25
Controls 26
Level Design 27
Mechanics Analysis 30
Schedule 31
Budget 32
Change Log 33

Executive Summary

Evicted is a first-person shooter set in a modern day apartment overrun by demons. The player will have to fight his way out and find the source of the demons if he wants to live. The game plays like an old school shooter with health and ammo pickups. It will test the player's skill in being able to kill enemies and stay alive.

Overview

Evicted is not the first of its kind, but that is not necessarily a bad thing. Taking inspiration from games such as Painkiller, F.E.A.R, and Nation Red, Evicted sets out to bring back the horror shooting genre using an old school feel. The game begins in an abandoned apartment complex that has been overrun by demons. The protagonist must battle his way to the source and send the demons back to hell.

Evicted's primary mechanics include shooting and level progression. The game spans 3 levels, going from apartment complex, to graveyard, to tomb. As the player progresses, the levels become more difficult via increased densities of enemies. He will have to refine his shooting skills as well as pick up more powerful guns along the way. The perceived difficulty will slowly rise with each level.

A variety of software was used to create the models, interface, and audio. Our content was designed in-house with a consistent overall mood and tone in mind. In particular, the Evicted team strove for a dark and chilling ambience intended to keep the player on the edge of his seat. Monsters were placed strategically to scream and jump out at unexpected moments.

Evicted can be played on PC, accompanied by a slightly stripped version for the Xbox 360.

Related Games

1. Painkiller Dreamcatcher / People Can Fly FPS 2004

Painkiller features fighting hordes of monsters in different arenas. The monsters were modeled after different types of undead, soldiers and criminals. Our game will differentiate itself from Painkiller by having scarier monsters that will try and panic the player. Painkiller also had a tarot card system where you could gain abilities by equipping the cards on your character before a mission. Our game will feature modern day first person shooter mechanics such as reloading and aiming down the iron sight of the gun.

2. F.E.A.R Monolith Productions / Vivendi Universal FPS 2005

Fear was unique by its dark ambient nature and its ability to build a certain level of suspense for players that constantly kept them on the edge of their seat. Evicted will have be similar in its dark and eerie nature, however it differs in the sense that we will be fighting off larger amounts of demons and will not use the bullet-time features found in F.E.A.R.

3. Nation Red DeizelPower Top Down Shooter 2009

Nation Red is a top-down twin stick arcade shooter featuring a protagonist who has to survive the zombie apocalypse. Waves of zombies are spawned all around him and the protagonist kills them accruing XP for each kill. After a certain amount of XP, the player is given a list of perks and can choose one to enhance his character. Our games will be similar because we will be fighting large amounts of enemies and using real world guns. They will differ by having different views of the action.

Player Composites

The Evicted team is targeting players who enjoy fast-paced action games with dark and chilling moods. Typically, our target players enjoy violent themes, thrill-seeking and mindless fun. We expect that these types of players will generally be males between the ages 13 and 27 who desire to alter their mental state after work or school. They are hardcode gamers whose interests include action-based movies and television shows with explosions, vulgarity and gore. Additionally, they likely listen to metal, classic rock and/or gangster rap music.

Our secondary audiences include the following: goth-enthusiasts who favor games and movies with a strong sense of evil and sinisterness; players who enjoy shooter-styled games primarily for the sake of shooting; and casual gamers who like to pass time without too much commitment from dropping out.

World

Evicted takes place current day on an Earth overrun by demons. The demons have risen from hell to cause chaos to mankind. Cities around the globe have fallen to their own local sources of evil, which tend to originate in nearby graveyards. The scope of the game will be at the protagonist's home, an apartment complex built near a tomb in Tomahawk, Wisconsin. It just so happens that the tomb contains one of the demonic sources of evil.

Characters

The Hero

In Evicted, you play as an ex-military veteran that is living a quiet life in Tomahawk, a small town in Wisconsin. You are a brave person who believes in justice. Having fought in three different wars, you are experienced for combat and know a good deal about first aid.

Being somewhat traumatized by what you have witnessed during the wars, your sleep is always interrupted by nightmares.

The Enemies

In Evicted, the enemies are demons that have started to invade our world. They are incarnations of pure evil that have no other motive than to corrupt, destroy and spread chaos. They have no will of their own, they simply act on impulse. During the game, the player will encounter 3 types of enemies.

The "Hellhound"

Two dogs, mid run animation

This creature was created from the stripped off flesh of other demons and molded together into a horrific dog-like shape that only knows pain, hunger and fear. Unleashed into our world, they seek out anything living and attempt to devour it.

Physical description:

- The dogs are not very big and are supposed to look like meat and flesh seared together.
- Always hungry, they have:
    - thin starving limbs
    - big stomachs in which to hold their food
- they have no eyes or ears (but their head has ear shaped flesh)

These enemies are the quickest in game. They usually scavenge and hunt in groups and will quickly swarm their prey. They will bite their targets to death and will devour them afterwards.

The dogs were meant to catch up to a running player fairly quickly. To counter this, they don't do much damage and they are not resistant to damage.

Stats:

- Health: 90
- Damage: 5
- Speed: 0.85

The "Demon"

Demon, mid run animation

These demons are the common folk from or dragged to Hell. Subdued to thousands of years of torture by horns implanted in their head, many have lost their limbs. Most commonly, they have lost their hands. Before the invasion, their demon overlords had them have hooks installed. The horns are also a way for higher demons to mentally influence the regular demons. Also, before they are sent outside of hell, they are dipped in a boiling hot vat of goo to encase them in skin and hung out to (painfully) dry like laundry on a line.

Physical description:

- The demons are formed from corrupted and captured people. They are of average human size
- Their skin is hard and thick to blunt damage
    - The skin covers the entire body so as to protect everything.
- Their hands are hooks so they can easily drag people back to hell for corrupting
- Curved horns in their heads for control and to hang them after a dipping

These monsters are the most common enemy in our game. They are the unit with the most health. However, they are also the slowest enemy type.

Stats:

- Health: 190
- Damage: 10
- Speed: 0.5

The "Fallen" / "Half Demon"

Half-demon, idle pose

These demons used to be angels that fell to the demon armies during combat. As a symbolic gesture, the demons cut the lower part of their body to keep them as a close to the ground as possible. To make sure that they stay low, they also cut their wings off and leave the saw implanted in them.

Physical description:

- Physically large (about 2-3x a normal human)
- Cut in half to show that they are further from heaven
- Skin ripped off
- Body parts replaced with demonic "augmentations"

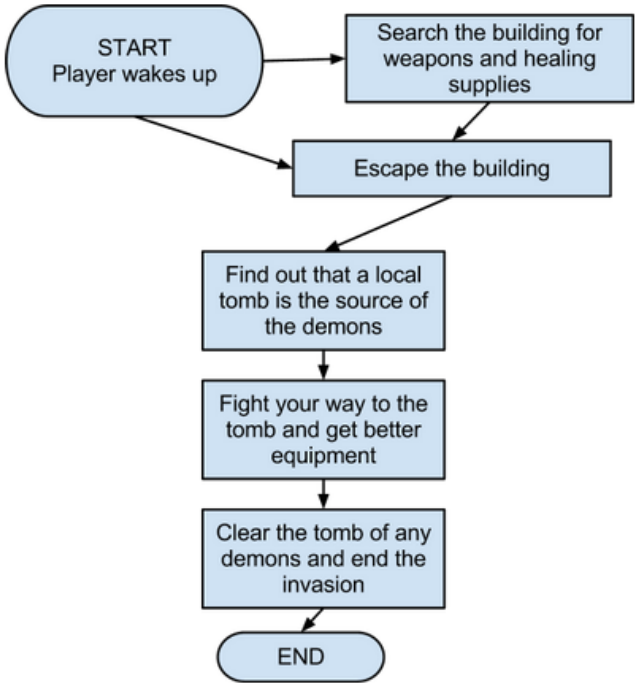
- Some of the angels are tough and get the saw blade stuck in them

These enemies are faster than the demon but slower than the dogs. They are the strongest hitting monster in the game.

Stats:

- Health: 100
- Damage: 15
- Speed: 0.75

Progression Graph



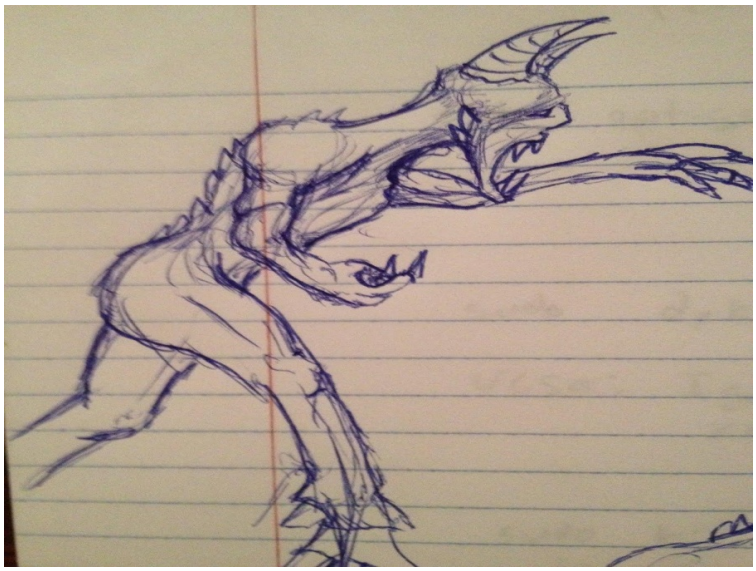
Art Direction

Concept Art

Demon - Crouched pose

Our original intention was to have the full demon be the final boss once the player had arrived at the tomb near the end of the game. We went for a dark theme, and wanted the demons to epitomize what hell may look like.

Demon - Side view attacking

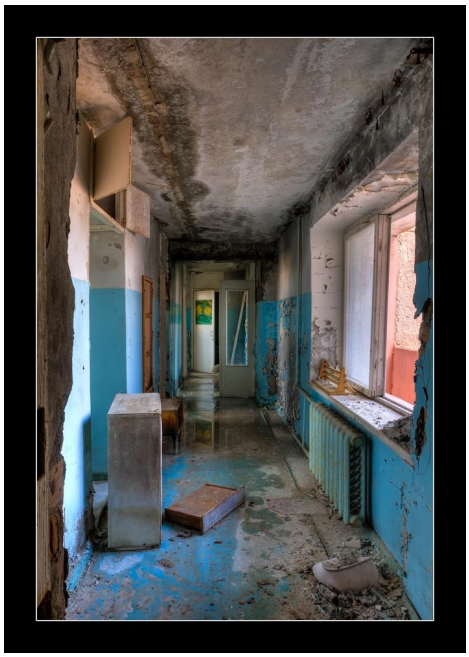


Half Demon Head

Reference Art/Inspirations

Abandoned Hallway

Used for setting ambiance and dark mood



Demon Dog

Referenced for 3D modeling in blender

Amnesia: The Dark Descent

The Shining (film)

Themes

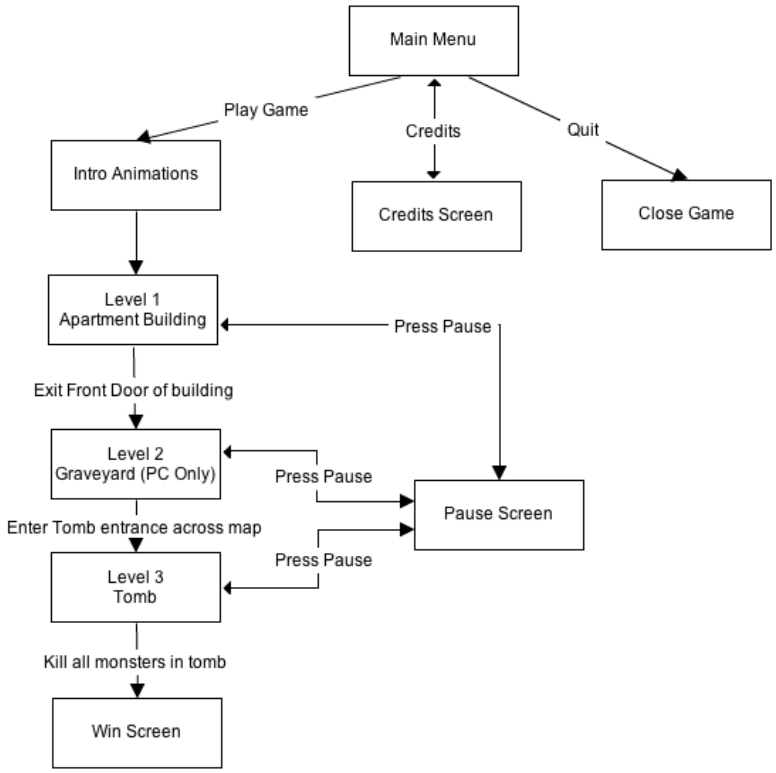
Evicted is meant to be a horror based first person shooter. We chose to stick to a dark ambience that was consistent with the feeling of a classic horror movie or other horror based games. The long narrow hallways in the apartment building allude to films like *The Shining,* Whereas the graveyard and tomb have a more classic doom reference. Within the tomb there are several references to pentagrams with the devils face on them, and refers to the biblical sense of demons.

Art Constraints from Technology

It was important to avoid too many polygons in models that may not be seen constantly by the player in game. Extras objects like Televisions, Desks, Showers, Toilets, we all modeled and placed in game, but we could not afford to make them too complex and use too many textures in order to keep a consistent frame-rate of 60 fps.

Time constraints also pressed us and did not give us time to fully implement the level of detail and diversity we had initially hoped for. We were planning to have a lot more explorable apartments, all being fully furnished.

UI Storyboards



Tags & Dialog

By monster type:

Dog, state:

- Agro1: Devil Dog Crazy.wav
- Attack1: Dog bitw.wav
- Die1: dog whine.wav

Demon, state:

- Agro1: Anger.wav
- Attack1: Stab.wav
- Die1: Stab.wav

Half-demon, state:

- Agro1: Strange growl.wav
- Attack1: Super punch.wav
- Die1: zombie long death.wav

Technology Plan

The game's engine was built from scratch by the Evicted team using Quake 3's level format (bsp) and a XNA version of the map loader. Modifications to the level library were made to update its code to XNA 4 as it had been developed for XNA 3. The Quake 3 engine has been open source for some time now under the GPL license.

The reason for developing the engine ourselves is because we did not find a better solution written for XNA. We worked on the engine components for rendering, physics and input.

Rendering

The rendering engine was made by the team and uses XNA's model renderer. A custom processor was created following a tutorial from a MS blog on how to import the model with the ability to play animations.

Level

Our maps are saved in the same format as Quake 3 (binary space partition). An XNA version of the loader is used to load them in. Modifications to the level library were made to update its code to XNA 4. The Quake 3 engine has been open source for some time now under the GPL license.

The team originally attempted to make the levels in Blender but ran into collision-related issues. Other options were explored, but the team eventually settled on BSP levels as one of the team members also had experience using one of its level editors.

Physics

The physics system is very basic and only employs gravity. Players and monster will drop if they walk off a ledge. A more complex physics system was not seen as necessary due to the scope of the game.

Collision Detection

Initially, the team was going to use GJK distance algorithm[1] to detect collision between convex meshes. This algorithm is very efficient and allows for very fast rejection of non-touching meshes. In addition, there is another algorithm that can use the intermediate information generated by GJK to generate a penetration vector which can be used to move a mesh so that it is not intersecting the with the base mesh.

However, due to time constraints and getting the models to correctly generate a convex mesh, it was decided to use XNA's axis-aligned bounding boxes. This also allows us to easily use XNA's built-in ray casting function to do hit detection.

Animation

One of the initial problems the team faced was playing multiple animations for one model. Due to a limitation in XNA, only one animation can be played from a FBX file. Thus, one solution is to have a model for each animation. However, that increases memory requirements for that file by N, where N is the number of animations.

Another option was to implement a MergeAnimationsProcessor[2]. This processor is a custom model processor that imports a model and references animations. The processor then merges them into one object. The referenced animations are then able to be applied to the model.

[1] http://en.wikipedia.org/wiki/Gilbert%E2%80%93Johnson%E2%80%93Keerthi_distance_algorithm

[2] http://blogs.msdn.com/b/shawnhar/archive/2010/06/18/merging-animation-files.aspx

Other tools

Programming

Visual Studio 2010 with the XNA Framework

2D art

Adobe Flash, Adobe Photoshop, Adobe Illustrator, Gimp

3D art

Blender

Music

Logic Pro, Massive, Adobe Soundbooth, Audacity

Level design

GTKRadiant

Repository management

Git on Bitbucket

Software Architecture

Core Classes

NightmareRun

NightmareRun is a child of the XNA Game class. It contains the majority of the general game logic, such as overrides of the Initialize, LoadContent, Update and Draw functions. Much of the programming for particular functionalities is outsourced to specialized classes discussed below.

Camera

The Camera class is used to encapsulate various data regarding camera position, orientation and movement. For example, it includes attributes for fovy, position and quaternion rotations.

Physics

The Physics class was implemented to handle any gravity and collision related logic. It's UpdateGravity method induces a negative Y force on an entity (discussed below). UpdateMonsterCollision responds to inter-monster collisions. UpdateCollision considers map-related collisions and updates an entity's directional velocity accordingly.

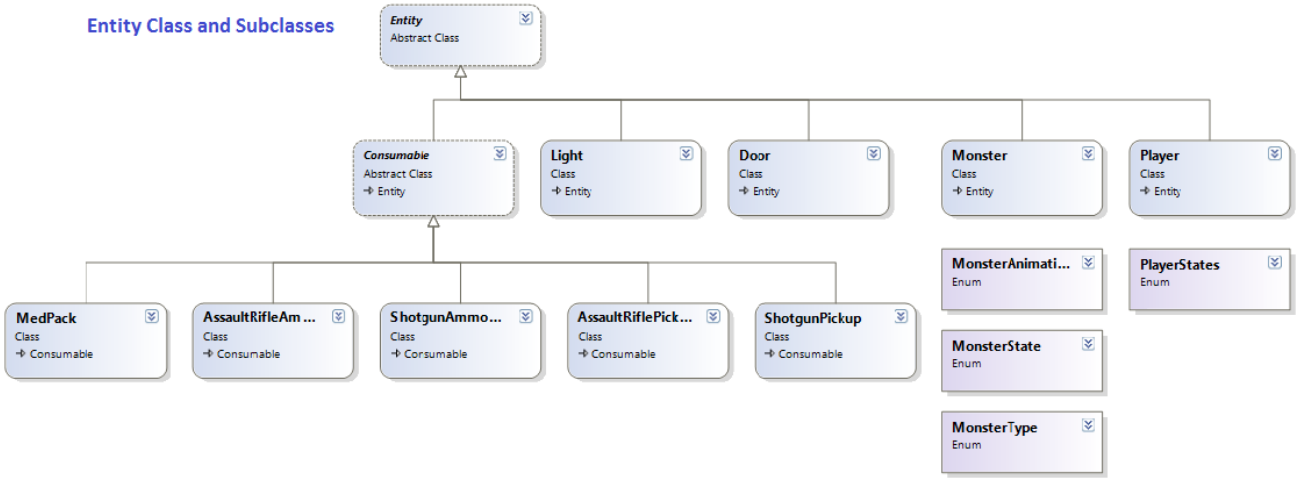
Input

Our Input class handles Keyboard, Mouse and GamePad input.

Renderer

The purpose of Renderer is to take care of drawing-related tasks. Levels, weapons and entities such as monsters and consumables are drawn in the Render function.

Entity Classes and Subclasses



Entity

Entity is an abstract class that encapsulates data regarding an object that might require physical and spatial qualities such as position, orientation and velocity. Additionally, an entity might have the capability for gravity and collision. Essentially, any physical object that could provide for some form of interaction in the game world could extend entity. Our Entity subclasses are Player, Monster, Consumable, Door and Light.

Player

This class is responsible for the actions of the player character. Since Evicted uses first-person perspective, player movement affects the camera directly. Thus, Player has a strong relationship with the Camera class. It includes methods for player movement and rotation, such as Move, RotatePitch and

RotateYaw. Additionally, it includes methods which interact with other entities such as OpenDoor and GetHit.

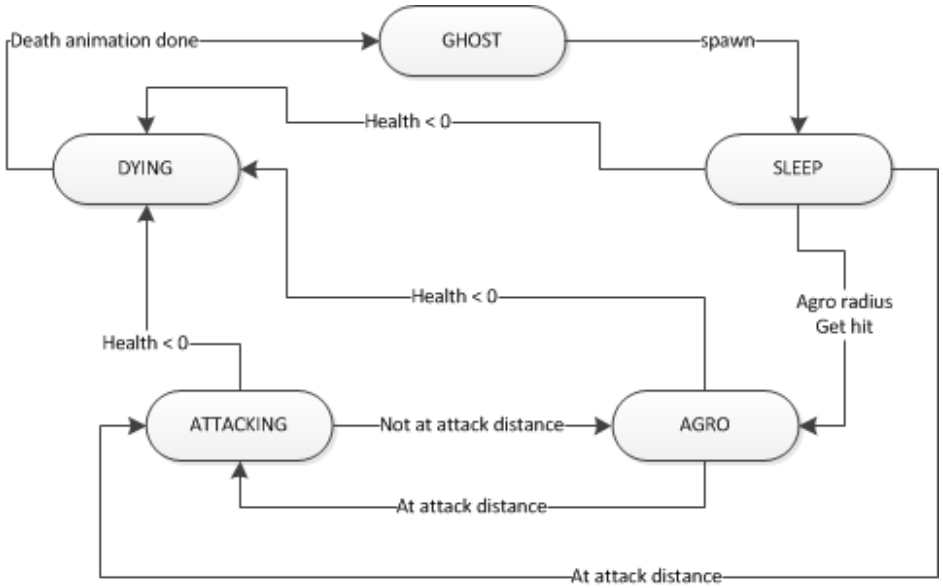
The Player states include no-clip, dead and alive. Dead and alive correspond to the health of a player and determine what a player can or cannot do in the game. No-clip was created for the purpose of debugging. It enabled the tester to travel freely throughout the map and walk through walls.

Monster

As one would expect, the Monster class is contains all the monster logic. Similar to Player, Monster has spatial functions such as updateDirection and updateVelocity. In addition to spatial tasks, the Monster class also deals with aspects such as animation and monster states.

Each monster has a type, state and current animation. The type determines the monster's qualities such as health, aggro radius and speed. Additionally, each type is associated with a monster character in particular. For example, DOG represents the Hellhound, whereas HALF_DEMON represents the Fallen. Documenting the type of monster is important as it determines the model used and its corresponding animations.

The actions of a monster are determined by its current state. There are five possible monster states: ghost, sleep, agro, attacking and dying. In the ghost, sleep and dying states, the monster is not moving or interacting with the player. Instead, it is sensing triggers to move onto another state. For example, if the monster is in the sleep state and senses the player nearby, his state will change to the agro state. In the agro state, the monster will follow the player around the map. In the attack state, it will attack the player. A representation of the state machine can be seen in the following flowchart.



The animations are also determined by the monster's current state. Each monster has four animations: static, walk, attack and die. When a monster's state is changed, its animation is also changed. Each corresponding animation can be seen in the table below.

State	Animation
Ghost	Not drawn
Sleep	Static
Agro	Walk
Attacking	Attack
Dying	Die

Consumable

Consumable is an abstract class used for items that the player can pick up. It contains an abstract method called ConsumedBy, which is called upon player collision. Each consumable subclass has its own purpose and model. The subclasses are Medpack, AssaultRifleAmmo, AssaultRiflePickup, ShotgunAmmo

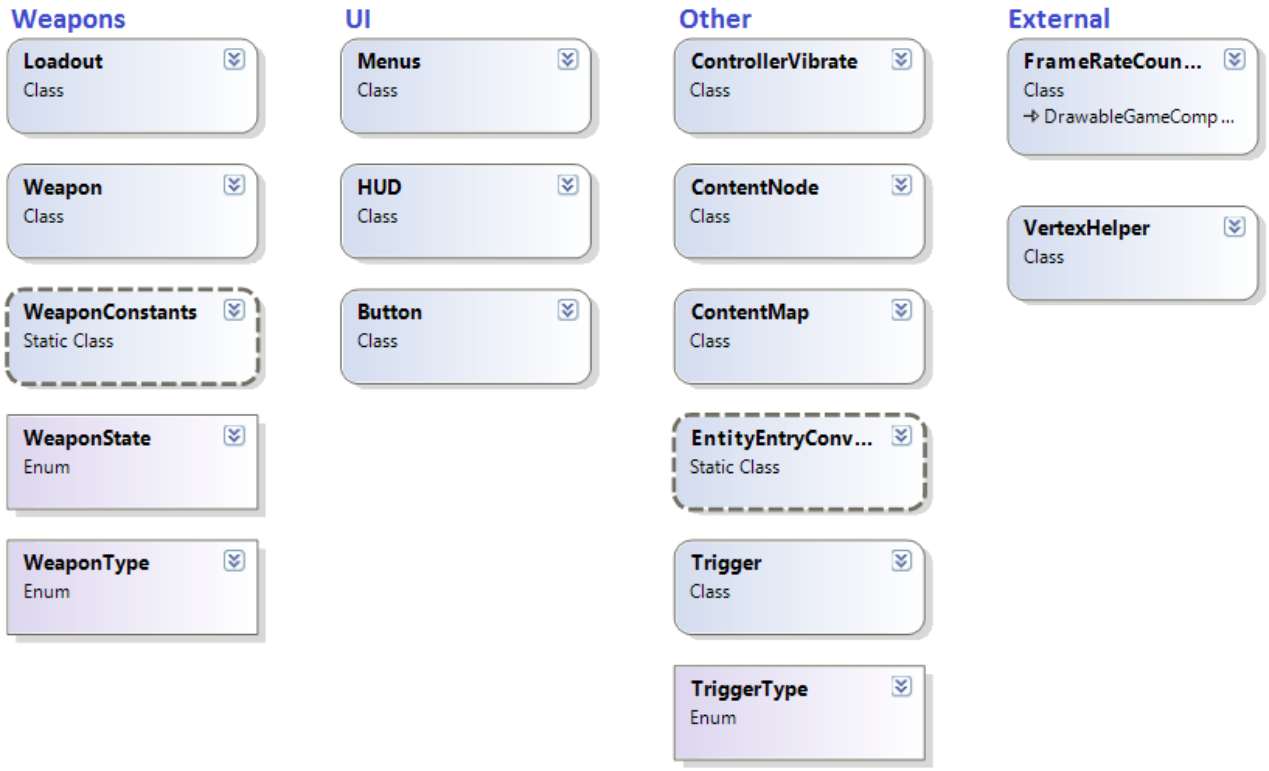
and ShotgunPickup. The medpack and ammo consumables increase the player's life and ammo. The "pickup" classes represent the picking up of a gun for the player's use.

Door

The Door class is a small class used specifically to open and close doors on the map.

Light

The Light class was originally created to provide for dynamic lighting. Each Light object would be used for its intensity and distance to monsters and guns. Lighting on each entity would then fade or strengthen dynamically. Midway through the project, we obtained functionality from this class. Unfortunately, due to performance issues, this class was deprecated.



Weapon-Related Classes

Weapon

The Weapon class encapsulates all gun-related attributes into one object. It includes attributes for damage, range, reload time, ammo, mag size, ammo reserve, screen translations and more. Some functions include reload, shoot and grabAmmo, which are used to update the weapon attributes accordingly.

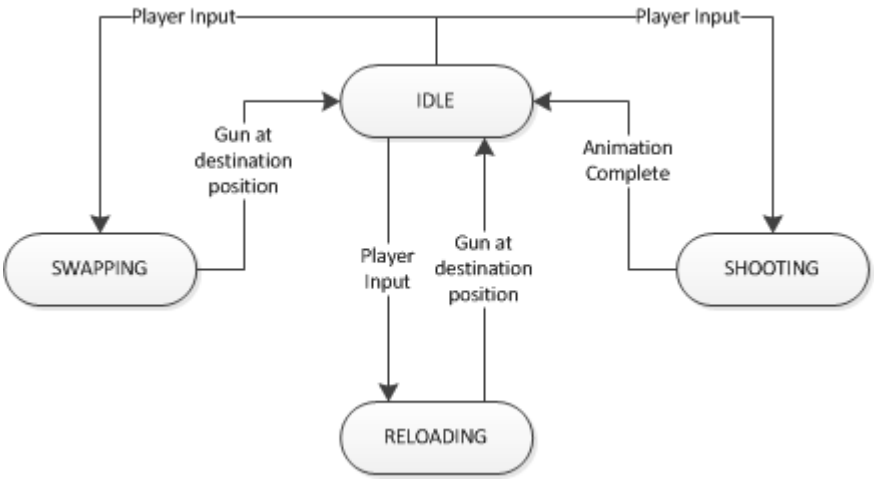
The current increment of Evicted contains three different guns (or Weapon Types): the glock, the assault rifle and the shotgun. These guns are created and held in the Loadout class discussed below.

Loadout

The Loadout class represents the player's inventory of guns and is used for gun interactions. At instantiation the Weapon Constants class is used to populate the Loadout's three guns with designer-balanced values. The player's current weapon is determined via the selectedWeapon attribute, which is an enum of WeaponType.

Similar to the Monster class, Loadout also has a state machine. Its states consist of the following WeaponStates: idle, shooting, reloading and swapping. These states are used to coordinate animations, sounds and interactions.

Each gun has two model animations: static and shooting. Animations for reloading and swapping were created programmatically by timing the lowering and raising of the guns. Each animation is commenced with each respective state change.



Loadout also contains the logic for shooting. Its shootWeapon function casts a ray down the player's line of sight. If the ray collides with a monster, we do additional testing to ensure the ray does not hit a map entity first and that the distance is appropriate according to the gun. If this is the case, then the victim gets hit accordingly.

UI-Related Classes

Menu

The Menu class is used as an introduction to the game. It handles all pre-gameplay menu navigation. It uses implementations of the *Button Class*, and loads the screens for the Main Menu, Credits. Also, it handles the introductory animations that lead to the gameplay.

HUD

This class is used to provide the player with important information regarding gameplay while they are navigating the maps. This includes the player's current health, how much ammo they have left, which weapon they have currently selected, and a reticle to help aim and attack at enemies. It also allows to display timed on screen messages that automatically disappear after a short period of time in order to provide the player with a sense of direction, or alert them about an item they have picked up.

Button

This is an external class used to represent clickable buttons for the game menu. This class was taken from Kaustubha Mendhurwar.

Other Classes

Controller Vibrate

This class handles making the controller vibrate due to event in the world, such as firing a gun or being hit by a monster. Everytime any other class wants to use force feedback, it calls this class which. In addition, this class handles multiple sources attempting to use force feedback and controls how long each pulse of feedback lasts.

ContentMap and ContentNode

These classes were created as helper classes for organizing models and animations. In particular, for monsters and guns, there are multiple animations for each type of a monster or gun. The ContentMap is a data structure used to easily retrieve each respective animation from a 2D array of ContentNodes. Content nodes hold animation related information, such as the model name and the animation clip name.

Trigger

This class is used to trigger level transitions. At map creation, trigger entities were placed and populated with a value representing the next map. During gameplay, if the player collides with the trigger object, then the next map is loaded.

EntityEntryConverter

This class was created as a helper class to convert Q3BSP data into data compatible with our program. As mentioned in our technology plan, our maps are created using GTKRadiant. During the map creation process, we create, define and position our own Q3BSPEntities as placeholders for our programmed entities. At level initialization, we read the list of Q3BSPEntities and convert them into our own programmed entities. The values from Q3BSPEntities are strings. The EntityEntryConverter converts the strings into types compatible with our objects. Additionally, the coordinate system is different in GTKRadiant than in XNA. This class is used to convert to our coordinate system.

External Classes

Vertex Helper[3]

This class was used to extract mesh information from the model file. This is was very useful because it allowed us to get the smallest and largest points in 3D space from a model. This allowed us to create better bounding boxes. We believe methods like these should be standard in a game development framework that already loads in model from files.

FrameRateCounter[4]

This class is used to display the frame rate in real-time through the application, as opposed to the console. This class was useful to us because our game is a 3D FPS as opposed to a turn based 2D RPG. It allowed us to check if our code was too intensive to maintain an acceptable level of responsiveness.

[3] http://www.discoverthat.co.uk/games/xna-vertexhelper.htm

[4] http://blogs.msdn.com/b/shawnhar/archive/2007/06/08/displaying-the-framerate.aspx

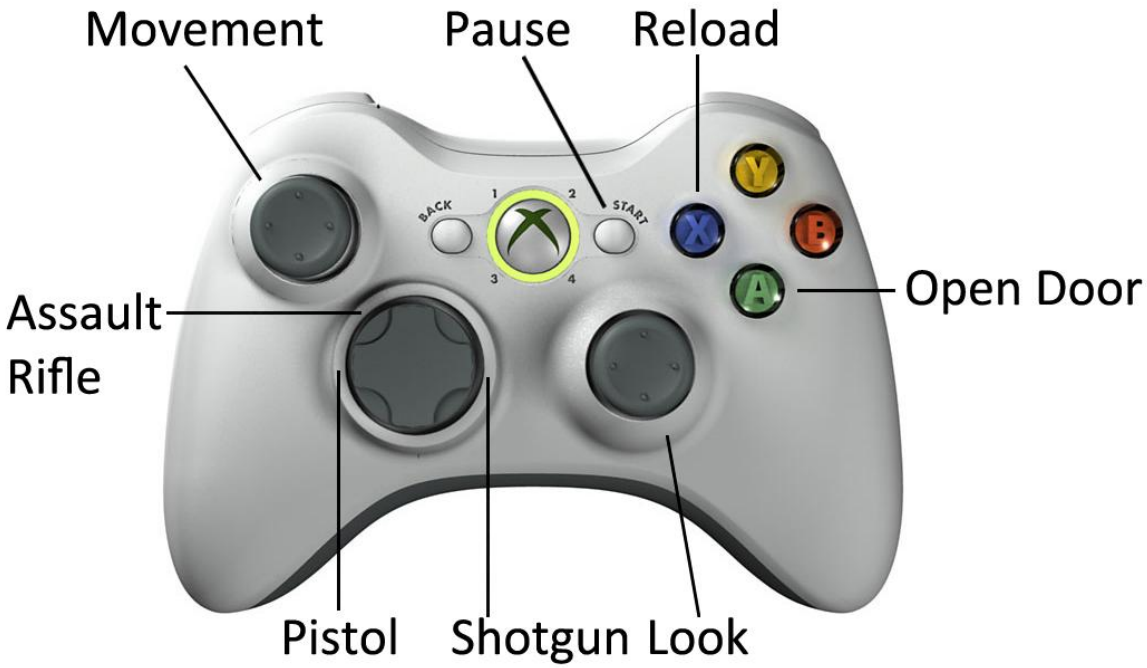
Controls

PC Controls

W: Move forwards
S: Move backwards
A: Strafe left
D: Strafe right
E: Open door
R: Reload

Mouse: Look up/down/left/right
Left Mouse Button: fire weapon
Right Mouse Button: Use iron sights

Xbox 360 Controls



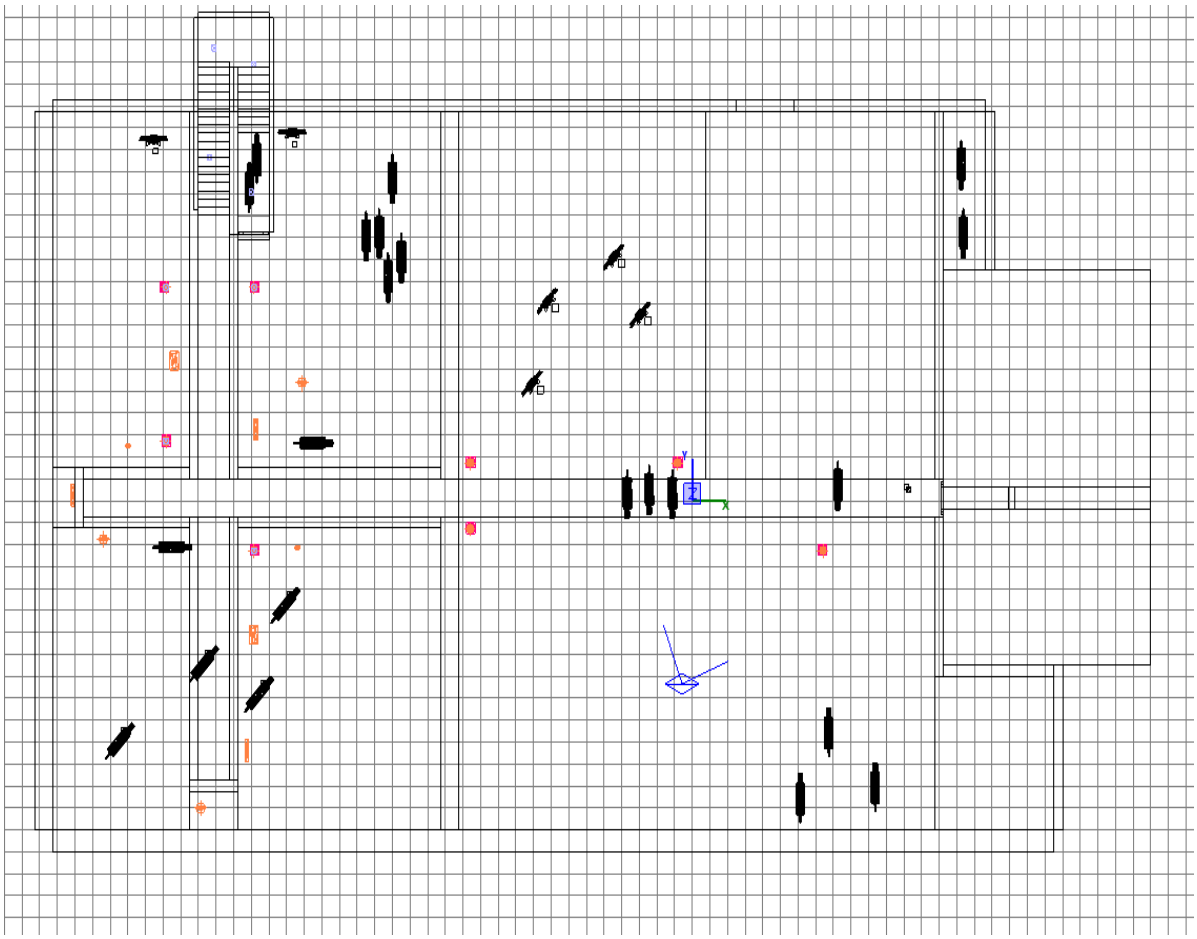
Level Design

Topdown blueprint of level 1 (apt_3) in GTKRadiant

The idea for the map is to give the player multiple routes to get to the exit. Hallways are narrow and apartments are small to give a claustrophobic feeling to the player. Because we felt like encouraging exploration, there is one room that med-pack and another that has the assault rifle pickup along with two ammo magazines to go with it. Most monsters are concentrated at the bottom of the apartment, near the exit to force the player to engage the monsters. The difficulty rating for this level is fairly low so that the player can get used to the mechanics and the controls.

Level statistics:

- **8 monsters (2 dogs and 6 demons)**
- **4 pickups (1 medpack, 2 assault rifle magazines, 1 assault rifle)**



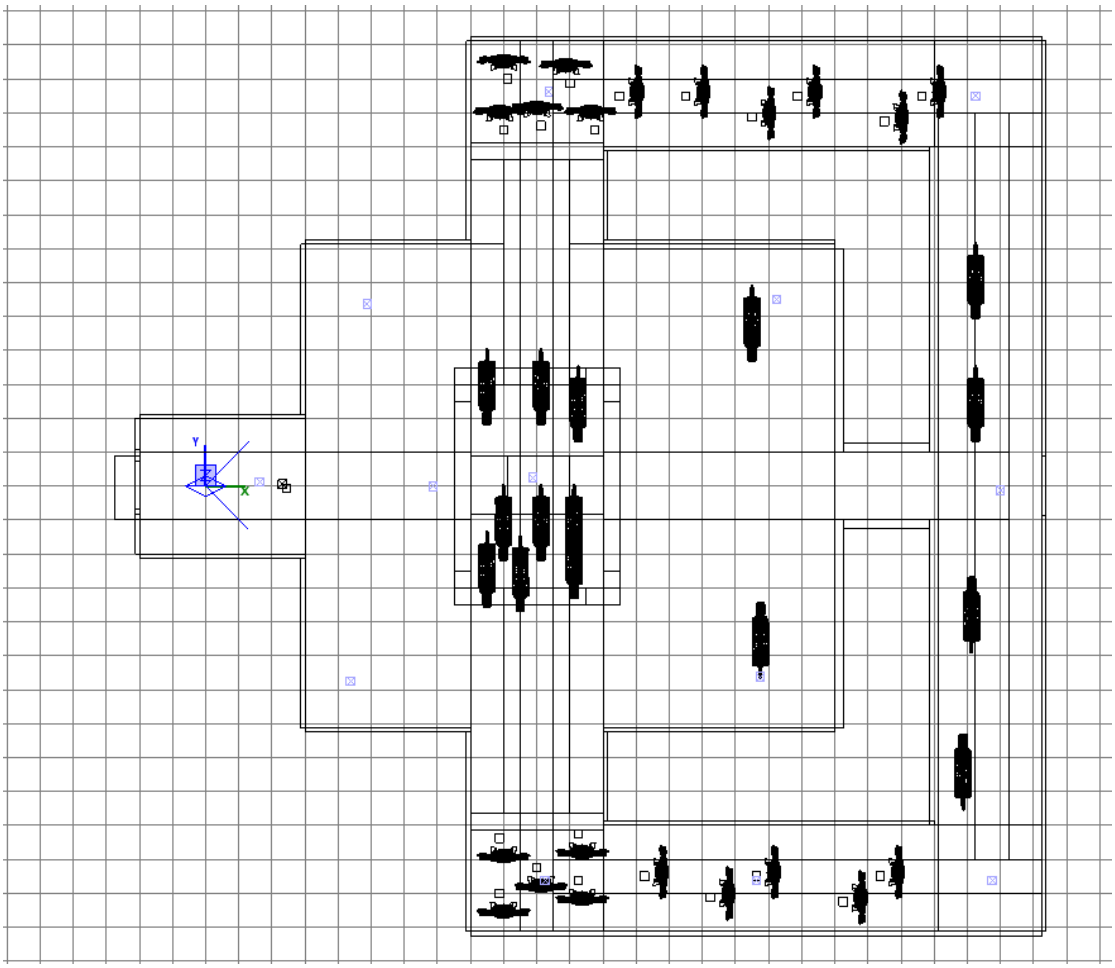
Topdown blueprint of level 2 (park_graveyard) in GTKRadiant

This map was designed to make the player feel uneasy. There is very little cover and the enemies are all around you. We expected the player not to stray away from the lit path, so we encouraged exploration by careful hiding two assault rifle magazines in a dark corner of the map. In this map, the player finds the last weapon, the shotgun. Also, this map is separated in two by brick walls. In the second part, the player finds a cemetery and gets ambushed by many enemies. Near the end of the level, down the stairs we placed two half demons, in order to stop the player from simply running to the exit.

This level was cut out of the Xbox360 version because it caused performance slowdowns.

Level statistics:

- **17 monsters (9 dogs, 6 demons amd 2 half demons)**
- **11 pickups (2 medpacks, 4 assault rifle magazines, 3 shotguns, 2 boxes of shotgun shells)**



Topdown blueprint of level 3 (tomb) in GTKRadiant

This map confines the player with a large amount of monsters. Because it is the last stage of our game, it is very challenging. In this map, there is no exit. Once every monster has been dealt with, the game ends. This level is more about surviving and dispatching enemies efficiently. The art style of the map lets the user on that he is getting closer to the source of the demon infestation.

Level statistics:

- **27 monsters (2 dogs, 21 demons amd 4 half demons)**
- **9 pickups (3 medpacks, 2 assault rifle magazines, 4 boxes of shotgun shells)**

Mechanics Analysis

First-Person Shooter

The primary mechanic in Evicted is shooting from first-person perspective. This mechanic requires quick reflexes and good aiming to hit fast-moving enemies before they can hit you. Killing the demons is required in order to win the game. Therefore, this mechanic is necessary to our game.

Player Skill

As there are many areas of condensed enemies in each level, player skill is necessary to get past such focus points. These areas will require a heightened sense of awareness and skill from the player in order to combat excessively large groups of fast-chasing enemies. This provides a challenging environment that we strive for in our action game. While killing enemies, the player will have to multitask and simultaneously avoid other enemies and obstacles. He will also be responsible for timing his reloads and gun swaps; and conserving scarce ammo for his stronger weapons.

Level Progression

Evicted for PC contains three diverse levels: an apartment building, a courtyard and a tomb. All three vary in monster quantity, key pickups and difficulty. Each level increases in difficulty from the previous level. This increased difficulty was designed at level creation by increasing the number of condensed enemies per square unit. Nonetheless, as the player progresses, his skill level will slowly increase from experience. In the first two levels, his power will also increase by obtaining better weapons. Thus, the players perceived difficulty will steadily throughout the game.

Level progression is also enhanced via checkpoints at the beginning of every level. When the player dies, he will be sent only to the beginning of his current level; not to the beginning of the game. Thus, he will not get penalized too much for each death.

Schedule

Week Of	Work
September 3	• Game conception
September 10	• Technology research (collision detection, map format/editor, viability on XBOX) • Infrastructure design discussions and brainstorming • Concept art drawn
September 17	• Entity and Player classes created • Renderer, Input, Physics classes started • Modeling commenced
September 24	• Proposal preparation
October 1	• Game proposal
October 8	• Weapon class started • Light class created
October 15	• Animations and texture-mapping started
October 22	• Monster class started • HUD designed and implemented
October 29	• Game progress report
November 5	• Loadout class started • Mapping commenced. BSPQ3Entity to Entity conversion logic created • EntityEntryConverter created
November 12	• Main Menu created • Animation logic and helper classes created • Renderer class completed
November 19	• Trigger class started • Door class created • Consumable classes created • Level transitions • Physics, Loadout and Monster logic completed
November 26	• Input class completed • Xbox deployment • Sound implemented • Haptics implemented • Evicted Increment 1 Complete

Budget

Our projected budget includes items such as per-hour wages, licenses for textures and software. Additionally, it includes goodwill towards open source software such as GTKRadiant, Audacity and Blender. Our models and most of our textures were created in-house. Therefore, their indirect costs are included within wages. The prices of priced software such as Microsoft Visual Studio and the Adobe Collection are not fully allocated to the Evicted project because they will be used for other future projects.

Item	**Cost**
Wages: (250 hrs * $20/hr * 4 workers)	$ 20,000
Texture Licenses: 25 * $4/texture	100
Visual Studio 2010 ($800 price * 4 licenses / 5 allocation)	640
Adobe Collection ($800 price * 2 licenses / 5 allocation)	320
Logic Pro ($200 price * 2 licenses / 5 allocation)	80
Massive ($200 price * 2 licenses / 5 allocation)	80
Goodwill	1,000
Total	**$ 22,220**

Under the assumption of a work-related project

Change Log

Week Of	Change Decision	Justification
October 29	1 Removal of perks	1 Projected time constraints
November 5	1 Removal of SpawnPoint class 2 Added one more monster	1 Placement of monsters during map creation process 2 Economies of scale
November 12	1 Added one more map	1 Economies of scale
November 19	1 Removal of Light class → Replaced with default lighting 2 Reticle flash	1 Performance issues 2 Enhance user interface
November 26	1 Simplified story 2 XBox: Removal of Level 2 3 Removed Key and Door progression 4 Added Monster counter 5 Did not implement audio as 3D	1 Time constraints 2 Performance issues 3 Time constraints 4 Provide information to player 5 Time constraints