Ubisoft NXT Programming Submission

By:

Asseel Sidique

➢ Personal Information

Full Name:	**Asseel Sidique**
School:	**York University**
Program:	**Computer Science**

➢ Biography

Born in Afghanistan, raised in Canada, I am aspiring to professional programmer working alongside AAA companies. I have started my programming during post-secondary studies at Humber College. During that period, I learned many programming languages, game design concepts and made several games. I am now working towards my Bachelor's Degree in Science specializing in Computer Science at York University.

➢ Resume & Cover Letter

Please see next pages.

ASSEEL SIDIQUE

1705 – 1155 Queen Street East, Brampton, ON, L6T 0G2
Home: **905-796-0631** | Mobile **647-526-1786**
asseelsidique@gmail.com

Professional Summary

I am a professional programmer in development with an emphasis on C++ with specialization in the development of video games.

Technical Skills

- Advanced experience with C/C++, Java, JavaScript and HTML/CSS
- Database programming using SQL and mysqlite3
- 3D Modelling in 3DSMax and Blender
- Creation of games through game engines including Unity, UE4, SDL, and Box2D
- Website development with WordPress
- Advanced experience with IDE's including Visual Studio, Android Studio, NetBeans, MonoDevelop and Eclipse
- Proficient understanding of Data Structures and Design Patterns
- Proficient understanding of Big Data fundamentals
- Experience managing small game development projects

Interpersonal Skills

- Advanced problem solving skills with object oriented solutions
- Excellent written and verbal communication skills
- Always ready to bring new ideas to the table
- Eager and willing to learn new software and programming languages

Education

2016 – Current	**Computer Science – Bachelor of Science (BSc)**
	York University

Relative Courses: Discrete Mathematics, Programming for Mobile Computing

2014-2016	**Advanced Diploma – Game Programming**
	Humber College

Relative Courses: Object-Oriented Analysis and Design, Data Structures and Design Patterns

References are available upon request.

ASSEEL SIDIQUE
1705 – 1155 Queen Street East, Brampton, ON, L6T 0G2
Home: **905-796-0631** | Mobile **647-526-1786**
asseelsidique@gmail.com

Ubisoft Recruiters

224 Wallace Ave #200

Toronto, ON M6H 1V7

Sunday, January 29, 2017

Dear Ubisoft Recruitment Team,

I'm pleased to have participated in Ubisoft's NXT Showcase in the programming discipline. Thank you for the opportunity! In this letter, I would like to demonstrate why I should be hired for an internship this summer and hopefully hired after graduation.

I am in my first year of studying Computer Science at York University and have already completed 2 years at Humber College for Game Programming. With my academic experience in college, I developed the skills to make several video games in different languages. Some of these languages include C/C++, Java and even shell scripting. My current portfolio which includes my work from college is displayed at http://www.sidique.000webhostapp.com. During the creation of these games, I became familiar with a variety of IDE's including Android Studio, Visual Studio, Eclipse, NetBeans and more. I have developed advanced skills working with game engines like Unity and Unreal Engine 4. I would love to develop my skills further by gaining experience in the field and I believe that is exactly what Ubisoft can offer me.

Aside from programming, I have great communications skills, work well with others and I am always looking for room to grow. I am looking to develop my programming career and stay with a company for many years down the road. I believe I can be successful as a programmer apprentice at Ubisoft because of my passion for programming and video games. Speaking of video games, some of my absolute favorite titles from Ubisoft are: Prince of Persia: the entire series (the best!), Splinter Cell and the entire Rainbow Six series. I am looking forward to being a part of the team that creates these prestigious games and seeing how I can help develop them with my programming toolkit and creative mind.

Thank you for giving us the opportunity and I hope to hear back from you. This apprenticeship is priceless and I am willing to prove myself worthy for it!

Sincerely,

Asseel Sidique

[illegible]

➢ UML and Feature Set, Code Description

Figure 1 shows a basic UML diagram of the code structure of the main classes. The code will be further described below.

The game structure is based on scene management. There is a main scene manager that handles all the different scenes (Game, Menu and Game Over). The scene manager's main role is to call updates to the current running scene and pass any information over to other scenes if needed (such as settings like turning the mouse on/ off and difficulty).

The game scene handles all of the actual game play. Objects such as the player, enemy, bullet and heads up display elements were created but were not directly instantiated into the scene (except for the player). Instead, a management system was put in place for each. For an example, the BMS or Bullet Management System was responsible for instantiation all of the bullets in the scene. The BMS used a vector to hold all of the bullets in the scene and destroy them once their life timer was equal to zero. The BMS also changes if the player picks up a shooting-related pickup. The enemies, heads-up-display, and background scenery all of a management system that wraps it.

Inheritance is used in this code structure, specifically in the Scene Manager class and the Player and Enemy class. The Scene Manager can handle all types of scenes as long as the derive from the Scene base class.

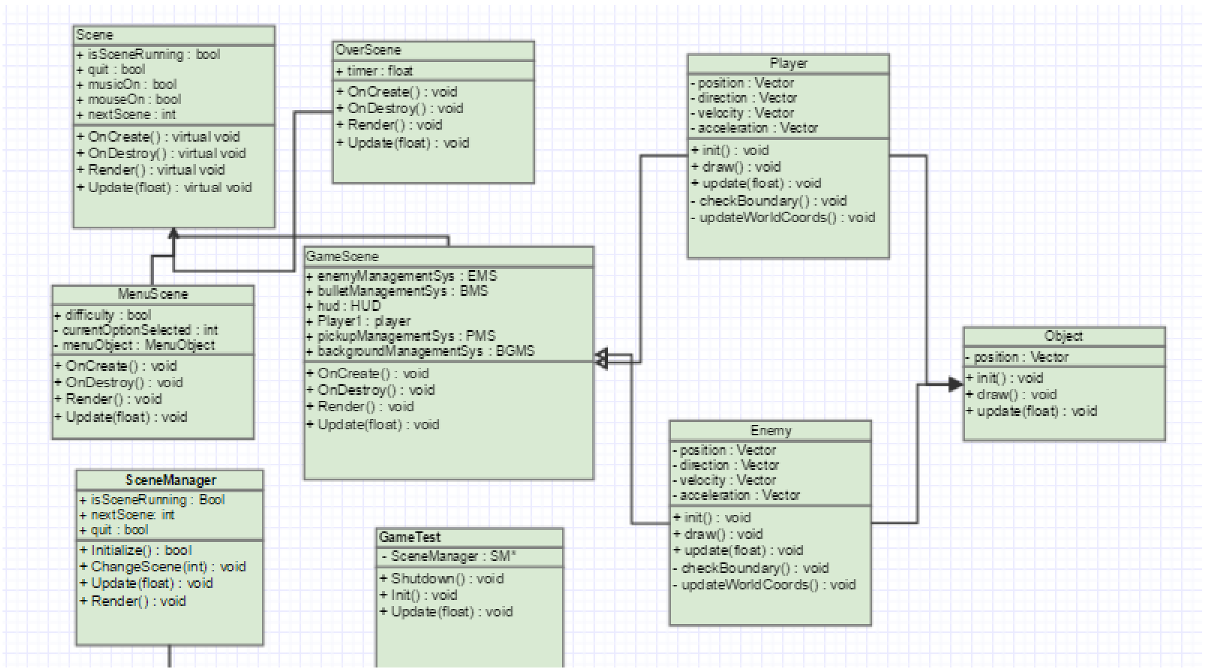

Figure 1

❖ PMS (Pickup Management System) and Upgrades

Upgrades and cash pickups were both treated the same in the PMS (Pickup Management System). The pickups are spawned and destroyed within the PMS. Each pickup has a life timer which runs out as soon as it spawns. The PMS won't spawn two or more of the same time of pickup at the same time.

The upgrades themselves were created by first detecting if the player is colliding with a pickup object. Then, inside the same block of code that detects the pickup, a method from the player class is called to add an upgrade to the player. The type of pickup is also passed through this call. Once passed, the player decides what attributes of its class needs to be changed based on the pickup. It also starts a timer that will reverse the changes of the pickup once that timer is set to zero. An example is the Rapid Fire Pickup. If this is picked up, the player knows to increase its Rate of Fire for 7.5 seconds and also to change the color of its body to yellow.

❖ BGMS (Background Management System)

The BGMS is responsible for the un-intractable objects that fly in the background and change color. This management system is comprised of a simple cloud object that draws a rectangle. There are two different types of clouds in the BGMS and one is larger than the other. The color of the clouds is also set to change in its update function based on taking the sin of some value that keeps changing. This value is then assigned to each of the arguments with respect to color in the DrawLine method. All of the objects inside the also has a certain scroll speed to which they move. This scroll speed can be assigned upon the creation of a cloud object as well as the size of that cloud.

❖ EMS (Enemy Management System)

The EMS is responsible for the asteroids that take up the screen. The EMS has a vector array of all of the enemies that are ever instantiated. The EMS keeps track of the round (since the number of enemies is based on the rounds). The EMS will produce asteroids and place them in random locations across the screen (making sure not to spawn right on top of the player). The EMS will then remove any enemies that are hit based on the collision calls from the game scene. When the remove method is called, the EMS checks first what phase the asteroid is. If the asteroid has a phase level that is greater than 2, it will not only destroy it, but create 2 miniature enemies in its place. If an asteroid was phase 3, the EMS would create 4 total smaller objects before that object is completely destroyed. This gives that effect of asteroids being destroyed to pieces! The EMS also handles a special type of collision which is the collision between two asteroids. If an asteroid collides with another, it will call the remove method for only one of the asteroids.

❖ BMS (Bullet Management System)

Some of this class was already discussed in the PMS section but it will be described in detail here. The BMS instantiates bullet objects every time the player pressed space in the game scene. The BMS keeps track of all the bullets alive and removes any bullets that run out of life (assuming the bullet is long gone). It also removes any bullets that collide with asteroids. The BMS will change its color based on the upgrade that the player obtains. The BMS also retrieves the rate of fire from the player in the game scene to determine how fast to shoot the bullets.

HUD

The HUD class is responsible for keeping track of the score, player lives and the asteroids left. This data is obtained from the respective classes in the game scene. The HUD also instantiates heart objects based on the number of lives the player has, up to a maximum of 5. The loseHeart method inside the HUD determines when to end the game as this method subtracts the amount of hearts on the HUD if the player collides with an asteroid. It can also add a heart if the player collides with a pickup. The HUD updates any changes to the score with a little +XX (XX = some number). This is called through the addScore method and takes in a Vector and Integer to determine where an asteroid was hit or a pickup was collected, and the value of the asteroid hit or pickup. Then it displays it to the screen for a short amount of time.

Round System

The round system runs inside of the EMS but is briefly described here. The player starts at round zero and makes their way up. The rounds are endless and will go until the computer crashes at infinity. After round 10, enemies speeds are increased by a big factor and it will get harder for the player to stop. Also after round 25, enemy phases are increased by 1. This makes enemies break for a total of up to 3 times and phase 3 enemies instantiate phase 2 enemies upon destruction. This means that phase 2 can still destruct further.

Menu

The main menu allows the player to customize two different options that will affect there gameplay. One of these is the difficulty which can only be switched to easy and hard. Secondly, the use of the mouse. If the mouse is on, the player will be able to use the mouse to look around and W/S keys to accelerate/slow down. The player will not be able to use the A/D keys to rotate when this option is turned on. If the mouse is off, the player will have full control of the ship using the WASD keys and space bar to shoot.

Vector class

A majority of the math involved in the making of this game used vector manipulation. The Vector.h class handled all of those manipulations with normalization, operator overloading, cross product and other useful methods.

LINK TO YOUTUBE VIDEO

Extreme Asteroids – Ubisoft NXT Programming Submission

https://www.youtube.com/watch?v=HU_0epbWnXg&feature=youtu.be