**Acknowledgement**

It is our greatest pleasure to take this opportunity to express our sincere gratitude and a deep sense of obligation to Mr. Daya Sagar Baral (Department of Electronics and Computer Engineering, central campus, Pulchowk Institute of Engineering) for providing us a creative space to build this project. We look forward to receiving his keen interest, guidance upgrading assistance, words of encouragement and valuable suggestions throughout the period of building this project. We would also like to express our appreciation to the Department of Electronics and Communication (DOECE), Center Campus, Pulchowk for providing us the platform to do this project.

Projects are integral to learning. They provide students a creative space to work and give life to their raw ideas. Projects implement the theoretical lessons learnt to a much more practical approach. Simply put, projects help us tackle real world problems in the future. Almost anyone can mug up the text, but it takes real understanding and implementation to day any project. So once again, we would like to thank DOECE for gibing s the platform to construct this project.

We are thankful and fortunate enough to get constant encouragement, support and guidance from all the seniors and teaching staffs which helped us in successfully completing our project work. Also, I would like to extend our sincere esteems to all staff in laboratory for their timely support.

**Abstract**

This project is a game made primarily on the concept of object oriented programming. It is a war game which consists of a plane which is assigned to destroy everything and anything within its sight. But of course, with war, there are always constraints and in this game, we have to destroy the objects as fast as possible because the fuel is running low.

**CONTENTS**

1. Objectives
2. Introduction
3. Application
4. Literature survey
5. Existing System
6. Methodology
7. Implementation
   1. Block diagram
8. Results
9. Problems Faced and solutions
10. Limitations and future enhancements
11. Conclusion and recommendations
12. References

**Objectives**

The main objectives which we hope we achieve at the completion of this project is listed below:

* Complete understanding and familiarization with object oriented programming, and its features such as inheritance, data encapsulation and polymorphism
* Basic understanding of what does behind the hood in simple games and their scope in the real world
* Learning to work in groups for a major project and being introduced to work division and working together for a final product.
* Using version control system like Git for tracking changes in project files an sharing code through “github.com”
* Using external libraries like SFML-2.51 for benefit and learning to implement the library functions in our program.
* Learning and familiarizing with basic concept of animation, collision detection of game object and implementing it on our project.
* Familiarization with 2D graphics in general and using it to make a two dimensional game
* Developing necessary sprites for the game on time and use it in our game environment
* Illustrious design of the game environment and the game objects in time for the project
* Applying minor game physics for throwing the projectiles out of the plane and also for the tanks and missiles
* Adding game feature like score sheet, audio effects, levels and so on.
* Applying the theoretical concepts that we learnt in the lectures of object oriented programming to build a project which can be useful in the real world application.

**Introduction**

In this our project, we plan to make a game. Our game revolves around a bomber plane which continuously flies in the sky. The ground at the bottom would consist of game objects such as houses and military buildings. The objective of the game is simple. We have to destroy the game objects by throwing projectile targets at them. In time, as the difficulty increases, we plan to make moving game objects such as tanks which can hit back at the plane. We included unlimited no of missiles. The objective being to dodge bullets and survive as long as you can.

Speaking more about the game, it consists of the concept of "lives" in games. So the user will have be given a chance so as to be given a chance not to reiterate their mistakes. High scores may also be an added feature. We also plan to introduce the concept of "bonus packages", which will include features such as higher damage, nuclear holocaust, etc.

For now, our game will use keyboard as primary device for moving around the plane, but may also use the mouse in some cases. The moving game objects will move around on their own with a constant speed. But we also plan to include these moving objects such as tanks into this game. So, in a multiplayer feature in the distant future, another user will also be able to throw bombs or bullets at the plane.

This game we plan to use the SFML library, having SDL as an alternative. We require sprites, animation and collision detection among other features and our initial survey has led us to the conclusion that our game will be well implemented if we use SFML library.

Our project is simple yet ambitious. It dwells around simple concepts of firing, dodging and surviving. Yet, it is ambitious in implementation and development in general. We are certain that this will be an addictive and fun game and we plan to begin development as soon as we get accepted.

**Existing System**

The closest similarity to our project is a game in the android Play Store calledAtomic Bomber which is developed by Luke Allen. In Atomic Bomber, you control a NATO ground-attack plane during a hypothetical 1970 Soviet invasion of Europe. You can drop both conventional and nuclear bombs. According to Tom Clancy, the NATO battle plan was to usetactical nukes during a Soviet invasion because the Soviet ground forces massively outnumbered NATO. In the game, you face Soviet anti-aircraft artillery (the ZSU- 57), SAMs (the SA-2), and fighters (the MiG-21). To keep the game interesting, you have to just avoid the fighters; they're invincible. In reality, (also according to Tom Clancy), NATO expected to have air superiority during a Soviet invasion, so the Soviet fighter threat in the game is probably unrealistic. Based on the high scores that the game submits to the web server, as of February2013, 82 million games of Atomic Bomber have been played since its September 2010 release, and roughly 495 years have been spent in game play. Recently an extra game mode has been added to the full version, where you can fly an A-10 Warthog armed with a GAU-8 30mm depleted-uranium cannon, plus Hydra 70 unguided air-to-ground rockets. The A-10's cannon can be used to shoot down the MiG (which was highly requested by players.) The MiG also gets shot down if a stray SAM hits it, in both game modes. The A-10 mode uses a separate online high score list from the regular mode. This was a highly successful game for the graphics it presents. We believe that we can make a game which is known more for its game play rather than its extra-essentials. The author claims to be porting his game for IOS in the future. As we can see, this game represents a lot of what we are going to do. But we do believe that our extra features, such as the extra game mode where one can also play as the tank, will improve the game for all of its legacy.

As this may be our core concept but we decide to recall the World War II, where japan was attack with the atomic bomb. Which destroyed Herosima and Nagasaki. The plane that throw that bomb was Boeing B-29 which is our plane. For our project plane is going to throw bombs to tanks that are auto operate on ground.

**Literature Survey**

Since this project is based on the application of object oriented programming concept, different books were suggested to refer by the teachers for the OOP concept. The books like “The secrets of object oriented programming” were referred OOP as reference for the development of program that assisted us to clear the concepts regarding the language and make our program development easier. For implementing the graphics in our project we have use SFML and for this, different e-books were referred and different video tutorials from the website were viewed.

**References**

Most of the coding involved in this project was because of our knowledge of object oriented programming derived from the lectures attained and lab sheets prepared for our class. Besides, we referred various books and websites to be familiar with and gain knowledge of some specific functions and features of C++ Programming. Moreover, we also took the help of seniors in some portions of our program. The following are some of the references which contributed to completing the project:

**Books:**

1. “SFML Game Development” Jan Haller, Henrik Vogelius and Hansson
2. “SFML Game Development By Example” Raimondas Pupius
3. “The Secrets of Object Oriented Programming in C++” Daya Sagar Baral and Diwakar Baral

**Websites:**

For the version Control: <https://www.github.com>

For video tutorials: <https://www.youtube.com>

For solving errors: <https://www.stackoverflow.com>

For the communications: <https://www.discord.com>

For every search: <https://www.google.com>

For SFML documentation: <https://www.sfml-dev.org>

**Methodology**

For our project we used SFML library which is short for Simple and Fast Multimedia Library. SFML library in itself is a pretty vast library so we only required the graphics, system and audio library. Organizing a project can be pretty overwhelming and it was at times for us, so we decided to divide our project into major but vague divisions which had minute sub-divisions. The flow of the program, we decided, should be such that, the game wouldn't begin straight away. The program flow is listed below.

1. **Splash Screen:** The game begins with a splash screen and an intro sound. The sound would be befitting of the game. We decided to build the splash screen ourselves but the sound effects were certainly extracted from an open source market. The Splash screen displays the title of the game and also the names of those who contributed to making it.
2. **Main Menu:** The main menu is simply and interface before the start of the game. In the main menu, you can start a new game, see the high scores or quite simply exit the game. The user arrives at the main menu at the start of the game/program, after the game is over, or after viewing the high scores.
3. **Start Game:** In the start of almost every game, there is always a game loop and ours is no exception. There are few things simultaneously happening in the game loop.
   1. **Drawing and movement of game objects:** In each frame of the game loop we clear and draw everything from the background of game, to the game objects. The game objects are constantly moving and each game objects have their own speed. For instance, the plane has more speed than the tanks. So each, time the game loop continues, the offset of plane increases more than the tank. Which is exactly how they move, or appear to move in the screen. The plane has the arrow keys controlling it's movement. Missile is launched by the press of a keyboard key which goes down in a projectile motion. The tanks have a fixed movement.
   2. **Collision Detection:** In each frame, we check whether the missile launched from the plane has collided with the game objects or not. For this, the missile should not necessarily hit the center or origin of the game object, rather it can collide within a tolerable proximity of the game object and still destroy it.
   3. **Explosion:** Explosion and collision are related terms and hence, the action of one triggers another in our game. We have prepared an animated explosion from a sprite sheet when collision between game objects is detected.
   4. **Fuel:** The fuel factor is how the game terminates. The fuel is continuously decreasing but if you get a hit, the fuel increases ever so slightly. So user has to hit targets with precision to survive.
4. **High Scores:** You get a high score depending upon the accuracy with which you threw those missiles. The high score is stored in a file and extracted when needed.
5. **Exit:** As the name suggests, you exit the game when you press this button.

**Implementation**

Start

Splash Screen

Main Menu

Exit Start Game

A

Escape pressed High Score High Score   
   
 Plane Fuel Depleted Game Object Collision

Redraw game Objects Score Counter

Exit

Explosion

Game Loop

Check Collision

High Scores