



# Level 100 Pikachu

Josh Jacobs

Jon Churchill

Chris Knight

## Table of Contents

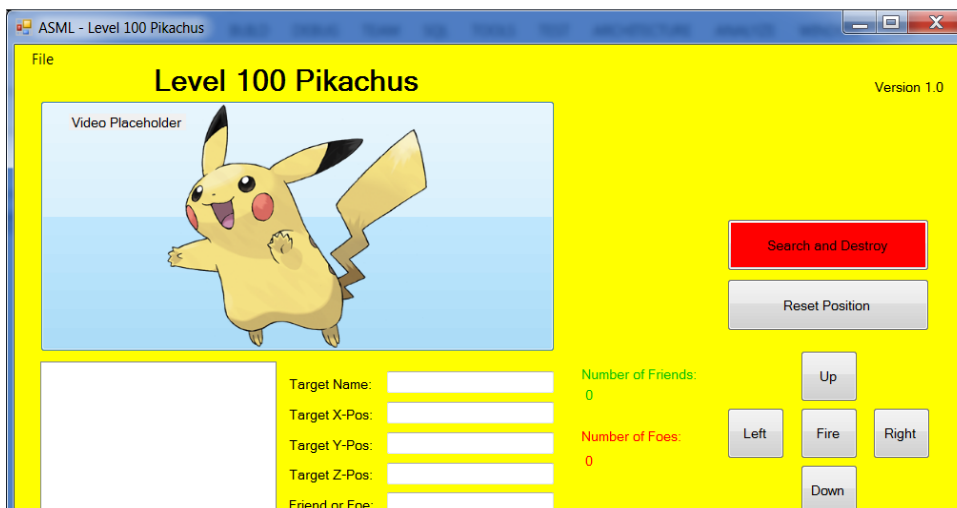
Description.....	2
GUI Storyboards.....	2
User Narratives.....	4
Formal Use Cases.....	4
UML Modeling Diagrams.....	8
Design Patterns Used .....	9
Design Considerations .....	10
Issues.....	11

## Write Up

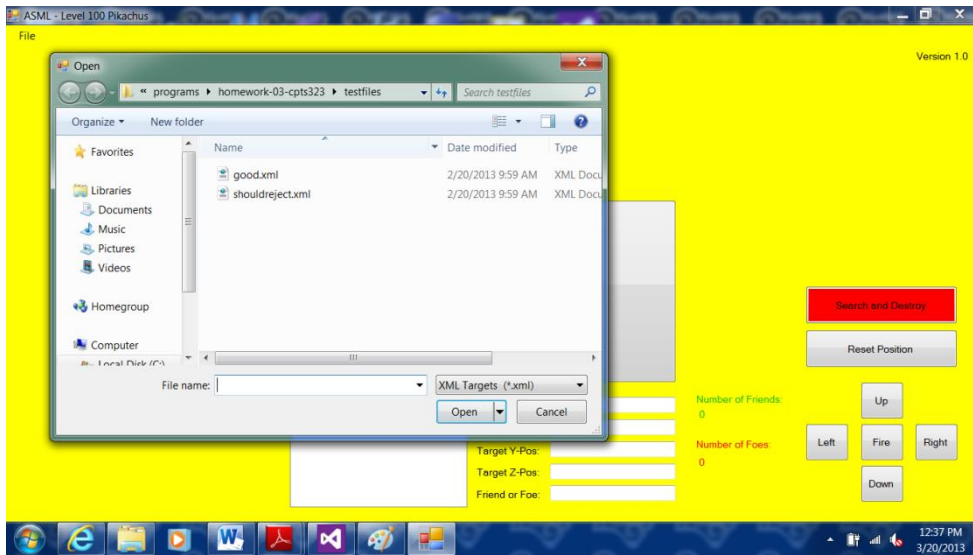
### 1) Project Description

This project is to create a graphical user interface (GUI) to control a dream cheeky Thunder missile launcher. The GUI will be able to load targets from an “.xml” or “.ini” file if the format is valid. These targets will then be loaded into the GUI and displayed in a list box. The GUI will have clickable buttons for directional movement of the missile launcher, and a button for firing missiles. When either the up, down, left, or right button is pressed, it can be held down to increase acceleration of movement. When the button is released, the movement will stop. When the fire button is pressed, a missile is fired from the launcher, and a new missile is put into the correct firing position, so when fire is pressed again the next missile is launched. When the user selects a new file from within the GUI, the previous target list will be removed, and a new target list will be loaded in. The GUI also contains buttons for search and destroy, and a place holder for video feed. These two functions have not been implemented for this stage of the project, but are set up for the next project.

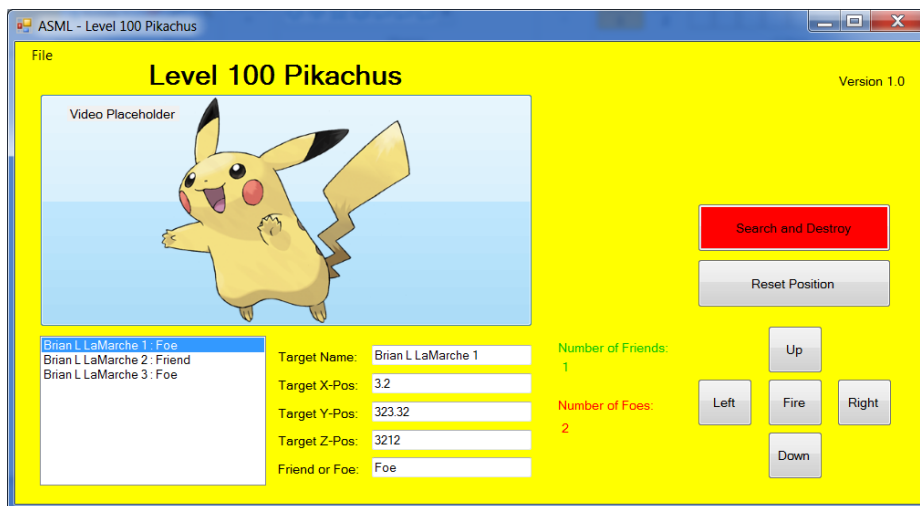
### 2) GUI storyboards



When the GUI is started up, this will be the first screen that the user will see.



When choosing to open a file, the user will be presented with a pop up menu that will allow them to choose either xml or ini files.



Once a file is loaded, the target list box will be updated, and the target information will be displayed just to the right of the list box. Also the Number of friends and foes will update.

When a fire or directional button is pressed, nothing on the GUI will pop up, but the launcher will follow those commands correctly.

### 3) User Narratives

Judge is a long time programming professor that wants to see some real world applications with some of this pet projects. With some code snippets he gave to all of his students he wants to see how well they can implement his rocket launching software to target and fire on only red blinking target cut outs.

Chris, Josh, and Jon are preparing to get the ASML ready for the competition. They need to make sure that the manual controls for the launcher work, and can be controlled through a GUI. They want to be able to move the launcher and then fire missiles at targets.

At the day of the launcher competition, the Judge wants to be able to load a target file into the GUI and see the target information displayed. He wants this as hassle free as possible when moving from event to event. He would like to be able to load as many files as he wants, each one presenting a brand new set of targets, not caring about the previous ones.

While testing the ASML aiming mechanisms, Chris, Josh, and Jon become impatient by clicking the movement button multiple times every time they move the launcher. Being lazy and not wanting to click that many times, they would like to be able to hold the button down and have the movement speed up as it goes.

### 4) Formal Use Cases

ID:	UC-1
Title:	Firing Missiles
Description:	When the fire button is pressed on the GUI, the ASML should fire a foam missile.
Primary Actor:	Chris,Josh,Jon
Preconditions:	The GUI has been started but is not in the search and destroy mode, but a mode that allows for the use of manual controls.
Post conditions:	A single missile is fired.
Main Success Scenario:	When the fire button on the GUI is pressed, the missile launcher is sent a signal to fire, when this happens, a single missile is fired and the next missile will be ready to be fired, for when the button is pressed again.
Extensions/Alt paths:	1: When the fire button is pressed, the launcher doesn't fire 2: When the launcher fires, it shoots all its missiles, not just one 3: When the launcher fires, it does not set itself up to fire again
Frequency of use:	Used for testing, and possible requirement for competition.
Owner:	Christopher Knight
Priority:	Infinite, which is our first priority
Risk:	Mit_04

ID:	UC-2
Title:	Loading file from GUI
Description:	Once the GUI has been started, a file needs to be chosen to load targets from. This will be selected from a supplied directory.
Primary Actor:	Judge
Preconditions:	The GUI has been loaded, and is sitting an idle mode.
Post conditions:	The file has been loaded, and the GUI can now display the targets.
Main Success Scenario:	When the user selects a file, there should be a choice of only .ini, or .xml. When a file is selected, the user should be taken back out to the GUI so they can perform other actions, and the target files should be read by the program.
Extensions/Alt paths:	1: When going to select a file, there is not one with the right extension for the user to choose. No file should be selected in this case, and the user should be returned to the GUI. 2: The user correctly chooses a file to read, but the file is a bad format so targets cannot be loaded. 3: After one file is loaded, another file needs to be opened with a new set of targets
Frequency of use:	Every time the GUI is used, and for backup if the camera fails
Owner:	Josh Jacobs
Priority:	100, second highest priority
Risk:	Mit_04

ID:	UC-3
Title:	GUI buttons for missile control
Description:	Once the GUI has been started, buttons for movement of the ASML will be active so that the launcher can be moved left or right, and be tilted up or down.
Primary Actor:	Chris, Josh, Jon
Preconditions:	The GUI has been loaded, and is sitting an idle mode.
Post conditions:	The launcher moves direction matching the button pushed.
Main Success Scenario:	When the user presses a directional button on the GUI, the launcher moves in that direction.

Extensions/Alt paths:	1: When a directional button is pushed, the launcher does not move, or react to the button signal. 2: When the button is held, the launcher moves, but does not accelerate. 3: When a directional button is pressed, the launcher moves, but doesn't stop moving when the button is released
Frequency of use:	The manual controls will be used for testing, but the actual movement of the launcher will be used every time the launcher is used.
Owner:	Christopher Knight
Priority:	Infinite, which is our first priority
Risk:	Mit_04

ID:	UC-4
Title:	Missile launcher speeds up
Description:	Once the GUI has been started, buttons for movement of the ASML will be active so that the launcher can be moved left or right, and be tilted up or down. When held, the launcher moves faster
Primary Actor:	Chris, Josh, Jon
Preconditions:	The GUI has been loaded, and is sitting in an idle mode.
Post conditions:	The launcher moves in the direction matching the button pushed, and speeds up when the button is held.
Main Success Scenario:	When the user presses a directional button on the GUI, the launcher moves in that direction. If the button is held down, the launcher should accelerate as long as the button is held.
Extensions/Alt paths:	1: When a directional button is pushed, the launcher does not move, or react to the button signal. 2: When the button is held, the launcher moves, but does not accelerate. 3: When a directional button is pressed, the launcher moves, but doesn't stop moving when the button is released
Frequency of use:	The manual controls will be used for testing, but the actual movement of the launcher will be used every time the launcher is

	used.
Owner:	Christopher Knight
Priority:	1, so it's a third priority, on the lower half
Risk:	Mit_04

ID:	UC-5
Title:	Program doesn't recognize USB launcher.
Description:	The GUI is running, but the application does not recognize that an USB ASML is connected.
Primary Actor:	Chris, Josh, Jon, or Judge
Preconditions:	The GUI has been loaded, and is sitting an idle mode.
Post conditions:	The ASML can be controlled with the use of the movement buttons from the GUI
Main Success Scenario:	The program recognizes the USB interface, and allows the user to control the ASML
Extensions/Alt paths:	1: The GUI doesn't recognize the launcher, and the GUI's buttons don't work
Frequency of use:	Every time the program is ran
Owner:	Jon Churchill
Priority:	1, so it's a third priority, on the lower half
Risk:	Mit_02

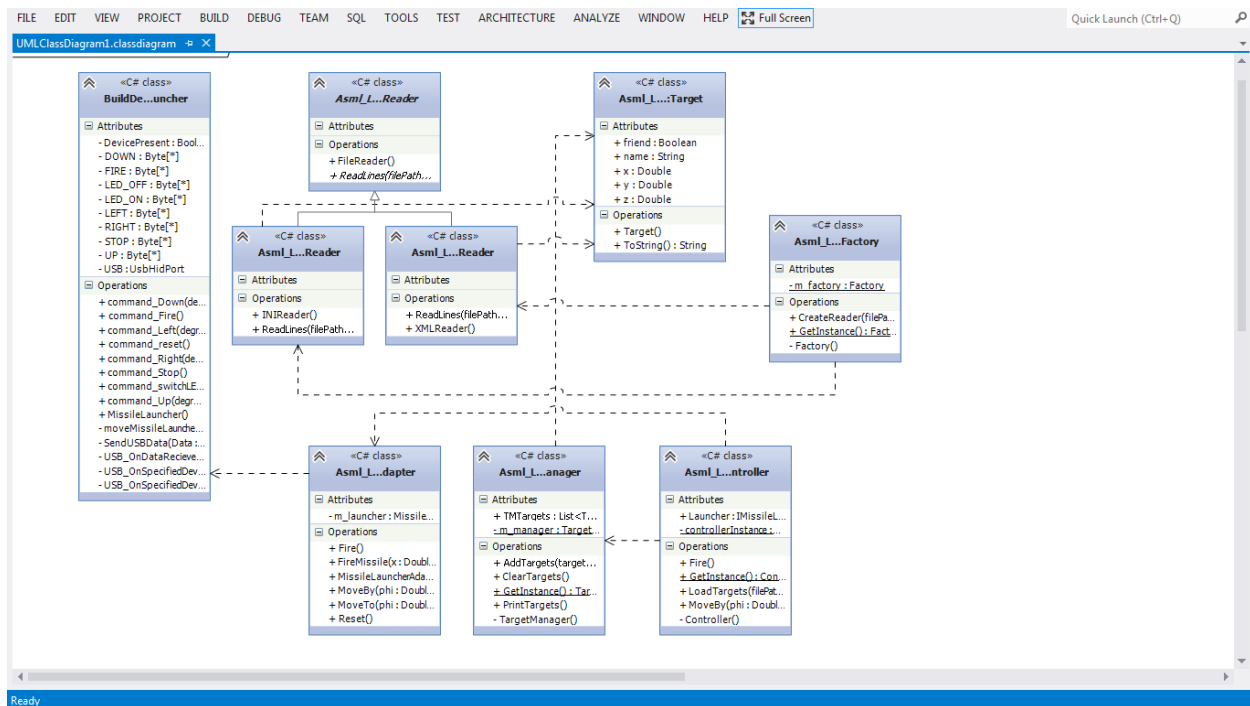
ID:	UC-6
Title:	GUI doesn't display proper information
Description:	The GUI is running, and it should display targets, team name, version number, and number of friends and foes.
Primary Actor:	Judge
Preconditions:	The GUI has been loaded, and is sitting an idle mode.
Post conditions:	The GUI displays all the vital information, and updates that information once a file is loaded.
Main Success Scenario:	When the GUI is started, correct information is displayed, and is updated as needed.
Extensions/Alt paths:	1: The GUI doesn't display team name or version number. These are hardcoded in so chances are unlikely of this happening
Frequency of use:	Every time the program is ran
Owner:	Josh Jacobs



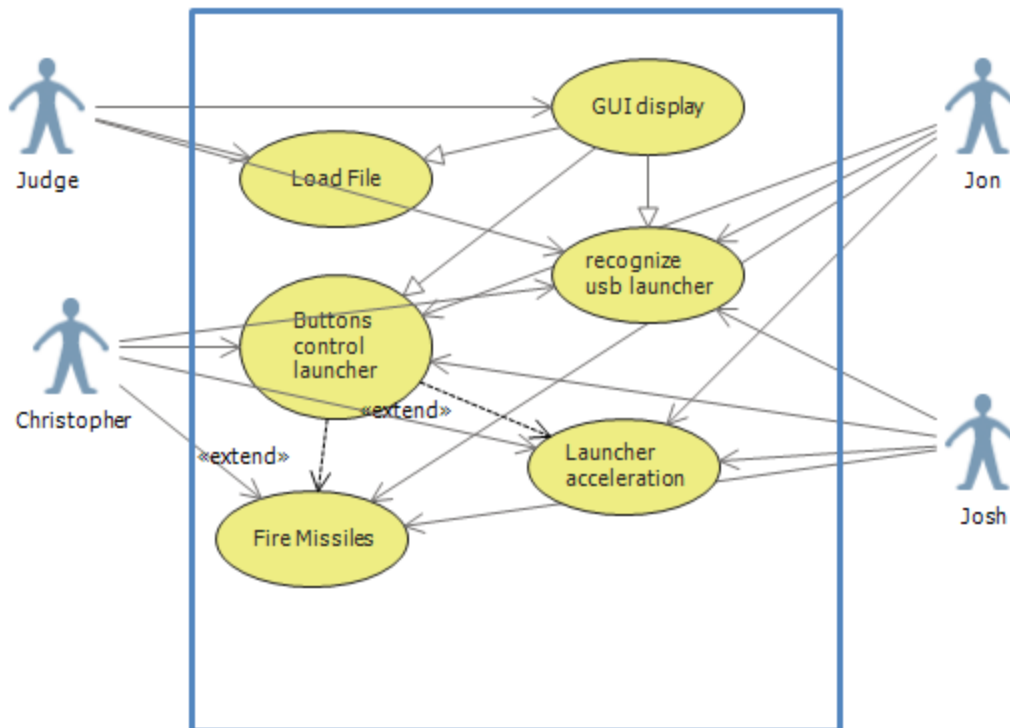
Priority:	Infinite, this is because it is top level GUI item that the rest of the program depends on
Risk:	Mit_06

## 5) UML Diagrams

### Class Diagram



### Use Case Diagram



## 6) Design Patterns used

The design patterns that we used were adapter, singleton, and factory. A Benefit of using the factory design pattern is that the file readers are not coupled. If later in the project, the requirements change, and we need to allow for another type of target file, the factory pattern allows for the easy insertion of another type of file reader. A good thing about using a singleton is when targets are read in from a file; only one set of targets can be active. When another file is read from, the original targets must be deleted to allow for a new set of targets. This ensures that we are not getting extra targets. The adapter pattern is very good for using the original launcher code, but not having to access it directly. The program that we use to fire, does not directly fire using the provided code. The adapter translates our fire to the original fire. This is good in case we get a new missile launcher class. We would just need to change the adapter to be able to talk to the new launcher file. A downside to using the singleton pattern for a target manager is if we wanted to read multiple files and keep appending targets onto

the list of targets. This is not allowed because only one set of targets can be loaded at a time with a singleton. A singleton launcher control is a good thing though so only one module can control the launcher. We do not want multiple modules trying to control the launcher at the same time.

## **7) Design Considerations**

The reason that we built the software the way we did was to keep the program from being tightly coupled, and adaptable to projects that are coming down the road. Using the adapter design pattern allows us to interact with the launcher without having to change any of the launchers code. When we say fire, the adapter knows how to talk to us, and then translate that to the missile launcher Google code without any problems. By doing this, if we got a new missile launcher source code file, we would only have to alter the adapter to be able to talk to it. The singleton design pattern makes sure that we only have one set of targets at a time. A consideration we had when using this pattern was when we were in the competition. We would be able to load a new set of target files for each event, clearing out the data from the last events file. The pattern also makes sure that there is only one missile launcher controller, so different commands can't be sent to the launcher at the same time.

## **8) Issues**

Whole Issue list

0 open issues

14 closed issues

Submitted

Updated

Comments

Reopen

Label

Assignee

Milestone

#14

Write-Up

Task

by glenn218k 6 days ago

#13

Tasks and Issue Tracking

Task

by glenn218k 6 days ago

#12

UML Class Diagrams

Task

by glenn218k 6 days ago

#11

Use Cases

Task

by glenn218k 6 days ago

#10

User Narratives

Task

by glenn218k 6 days ago

#9

GUI Screenshots

Task

by glenn218k 6 days ago

#8

Realize ILauncher Interface

Task

by glenn218k 6 days ago

#7

Design Patterns to Use

Task

by glenn218k 6 days ago

#6

Program Name

Task

by glenn218k 6 days ago

#5

Display Group Name

Task

by glenn218k 6 days ago

#4

Control Missile Launcher

Task

by glenn218k 6 days ago

#3

GUI Version

Task

by glenn218k 6 days ago

#2

GUI displays position/friend/name

Task

by glenn218k 6 days ago

#1

Read XML/INI targets

Task

by glenn218k 6 days ago



14 closed issues in this view


## Individual Issues


glenn218k opened this issue 6 days ago

Edit

### Read XML/INI targets

 glenn218k is assigned 


Milestone: **Project.3.21** 




The software package must be able to also read a target description file in XML or INI format.

Closed

0 comments

Labels 

Task

1 participant 

glenn218k opened this issue 6 days ago

Edit

### GUI displays position/friend/name

 knight360 is assigned 

Milestone: **Project.3.21** 






The GUI should display these targets with the following attributes: positions x,y,z; friend/foe; target name.


glenn218k opened this issue 6 days ago

Edit

### GUI Version

 professormurder is assigned 


Milestone: **Project.3.21** 





The GUI should display the version of the software at the top of the program.

Closed

0 comments

Labels 

Task

2 participants  

glenn218k opened this issue 6 days ago

Edit

### Control Missile Launcher

 knight360 is assigned 


Milestone: **Project.3.21** 



The missile launcher should be controllable from the GUI: move up/down/left/right; fire missiles.

Closed

0 comments

Labels 



Task

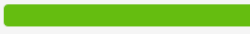
2 participants  


glenn218k opened this issue 6 days ago

Edit

### Display Group Name

 glenn218k is assigned 


Milestone: **Project.3.21** 



The GUI should display the group name.

Closed

0 comments




Labels 

Task

2 participants  

glenn218k opened this issue 6 days ago Edit


### Program Name

 glenn218k is assigned  Milestone: **Project.3.21**  



Your program should be called "Asml-Groupname.exe" where Group Name is your group's name.

**Closed**

0 comments





Labels 

Task

2 participants  

glenn218k opened this issue 6 days ago Edit

### Design Patterns to Use

 professormurder is assigned  Milestone: **Project.3.21**  

You must use the following design patterns: singleton; adapter; factory

**Closed**

0 comments





Labels 

Task

1 participant 

glenn218k opened this issue 6 days ago Edit


### Realize ILauncher Interface

 professormurder is assigned  Milestone: **Project.3.21**  


Your missile launcher should realize the ILauncher interface that is provided in the CptS323 GIT repository.

**Closed**

0 comments





Labels 

Task

1 participant 

glenn218k opened this issue 6 days ago Edit


### GUI Screenshots

 knight360 is assigned  Milestone: **Project.3.21**  



you must create GUI storyboards for each screen of the software. Each storyboard should show the different states of the GUI if the software enters different modes, e.g. search and destroy versus idle.

**Closed**

0 comments





Labels 

Task

2 participants  

glenn218k opened this issue 6 days ago Edit


### User Narratives

 professormurder is assigned  Milestone: **Project.3.21**  



Write a user narrative for each user in your system.

**Closed**

0 comments





Labels 

Task

2 participants  

glenn218k opened this issue 6 days ago Edit

## Use Cases





 knight360 is assigned  Milestone: **Project.3.21**  

use cases are required to model each user and system interaction.


1 participant 

glenn218k opened this issue 6 days ago Edit

## UML Class Diagrams





 glenn218k is assigned  Milestone: **Project.3.21**  

You must model your software with UML Class Diagrams. All classes need to be included in these models.

1 participant 

glenn218k opened this issue 6 days ago Edit

## Tasks and Issue Tracking





 glenn218k is assigned  Milestone: **Project.3.21**  

You must enter each development task into GitHub as an issue per direction of your software management plan that you turned in as a group.

1 participant 

glenn218k opened this issue 6 days ago Edit

## Write-Up

 knight360 is assigned  Milestone: **Project.3.21**  


Must include a write-up with the following sections:

- description
- gui storyboards
- user narratives
- formal use cases
- uml modeling diagrams
- design patterns used
- design considerations
- issues

1 participant 

Open

0 comments

Labels 

Task

Open

0 comments

Labels 

Task

Closed


0 comments

Labels 

Task

Open

0 comments

Labels 

Task