

# StarCraft Artificial Intelligence Player

Senior Capstone Project 2012-2013

Savannah Van Beek && Jonah Brooks

## **Table of Contents:**

Introduction	1
Original Requirements Document	2
Adjusted Requirements	9
Weekly Reports – P3's	10
Poster	21
Project Documentation	22
Technology Learned	38
What Was Learned	38
Savannah	38
Jonah	38
Appendix 1: Code Listings	39

## **Introduction:**

This StarCraft AI framework was requested by Dr. Alan Fern of Oregon State University. Over the summer prior to submitting this framework as a Capstone project, Dr. Fern worked with Jesse Hostetler, Joe Runde, and Jonah Brooks on developing a StarCraft AI player. His goal was to delve deeper into how Real Time Strategy game AI works in order to research many RTS, real world, and academic problems. Writing AI for real time strategy games offers many parallels to real world problems such as automated search and rescue and air traffic control, while also presenting many interesting challenges on an academic level. Over the summer, however, much of the group's time was spent implementing game tasks that are trivial for humans, such as placing buildings and moving units. At the conclusion of the summer project, Jonah Brooks met with Dr. Fern to discuss the possibility of creating a framework that would allow future students to skip over these rote tasks and delve directly into the more academically interesting aspects of writing an AI. After this meeting, Dr. Fern officially submitted the project for approval as a CS Capstone project.

Once the school year began, Jonah Brooks and Dr. Fern selected Savannah Van Beek as the second member of the team, and work began in earnest. Dr. Fern acted primarily as an advisor and supervisor over the project as a whole, while Jonah Brooks and Savannah Van Beek handled the programming, documentation, and design.

## **Original Requirements Document:**

### **Team Name:**

Educational StarCraft Artificial Intelligence Player (E.S.C.A.I.P pronounced Escape)

### **Team Members:**

Jonah Brooks -- [brookjon@onid.orst.edu](mailto:brookjon@onid.orst.edu)

Savannah Van Beek -- [vanbeeks@onid.orst.edu](mailto:vanbeeks@onid.orst.edu)

### **Client/Sponsor/Mentor:**

Alan Fern – [afern@eecs.oregonstate.edu](mailto:afern@eecs.oregonstate.edu)

### **Introduction to the problem:**

Alan Fern wanted to create a framework to facilitate teaching and researching artificial intelligence. The summer of 2012 Alan Fern lead a group of students to guide in the creation of a framework using the Brood War API. The general foundation has been laid, which gives us a starting point for the project.

### **Project Description:**

Our goal is to write a modular AI player framework that future students can use and modify. We will use this framework to create an AI player that is capable of playing as Terran on any map. This project would be considered a success if future students are able to use this framework to learn and expand upon our AI player implementation. We hope to get the framework to the point where we could submit our code into a StarCraft AI competition at the end of the year.

### **Requirements:**

This project was started over the summer by two interns as an undergraduate research project. The program language chosen was C++ and we will continue using C++ for convenience. The

other requirement for this project is the BWAPI (Brood War API). This allows us to program an AI for StarCraft.

Req			
#	Requirement	Status	Comments
1	Refactor Code	pending	
2	Initial Documentation	pending	
3	Evaluate Formations	pending	
4	Formation Development Tool	pending	
5	Integrate Formations	pending	
6	Example Formations	pending	
7	Evaluate Military Hierachy	pending	
8	Standard ArmyTask Template	pending	
9	Defensive ArmyTask Examples	pending	
10	Harrasment ArmyTask Examples	pending	
11	Unit Tests	pending	
12	Test Maps	pending	
13	AI Success Statistics	pending	
14	Example AI Scripts	pending	
15	Example AI Techniques	pending	
16	Start Version 4	pending	
17	Prepare for Expo	pending	
18	Expo	pending	
19	Final Report	pending	

## **Versions:**

### **Version 1:**

Here we will get the code ready for the project. There is a lot that has already been done but hasn't been documented, commented, or organized. There has been some code done by others that we may have to refactor in order to make it less race specific. Even though we plan on just using Terran our goal is to make it able to adapt easily to the expansion to the other races.

### **Version 2:**

At this point we will finish laying the foundation of the framework. We will then implement example Fireteams and ArmyTasks. Future students who are expanding on them will be able to use our examples to understand how they work and create their own.

### **Version 3:**

Once the foundation has been made and we have a handful of Fireteams and ArmyTasks we will create a test suite and a few simple AI scripts. We will use the test suite to verify that our code is doing what we think it should be doing, and the simple AI scripts will demonstrate how the AI will play at a basic level.

### **Version 4:**

This would be where we would create the actual AI. This is where the AI would determine what Fireteam and/ or ArmyTask would be best to use in certain scenarios. We would be optimizing existing code with the hope of creating something that has the potential to win. This is something we would like to get to, but only if time permits.

### **Design:**



### **Specific tasks to be undertaken:**

This is broken down into task and their subtasks. They are separated by version and then the order in which we will complete them.

#### **V.1**

1. Clean up existing code:
  - a. Refactor code to be more modular/ extensible
  - b. Document existing features

## V.2

1. Finish the formation system:
  - a. Evaluate existing structure
  - b. Make a development tool for easily adding new formations
  - c. Integrate formations into Fireteams and Squadrons
  - d. Create and document a few example formations
2. Formulate template and documentation for creating ArmyTasks and Fireteams:
  - a. Evaluate existing hierarchy for military
    - i. Focus on input/output for ease of use, similar to an API
  - b. Create a standardized approach for adding new tasks and Fireteams
3. Create and document new example ArmyTasks:
  - a. Create defense oriented ArmyTasks
  - b. Create harassment oriented ArmyTasks

## V.3

1. Create test suite:
  - a. Unit tests for each class/function
  - b. Test maps for stress testing AI and spotting AI regressions
  - c. Methods for tracking AI success rate over multiple games
2. Ensure that the framework is usable and flexible:
  - a. Make a few simple AI scripts that use the framework
  - b. Create a few simple AI's that simulate various AI writing techniques

#### V.4 (Optional)

Write a full Terran AI:

- Create a set of strategies, possibly extend into a suite for future use

- Create a simulator for estimating strategy success rates given game states

- Implement automatic strategy-switcher based on simulations

- Ensure the strategy switcher can win on given maps

- Ensure the strategy switcher can win on any map

- Tweak to improve success rate

Work on option add-on modules:

- Unit or formation micro with potential fields

- Zerg and Protoss support

#### **Risk Assessment:**

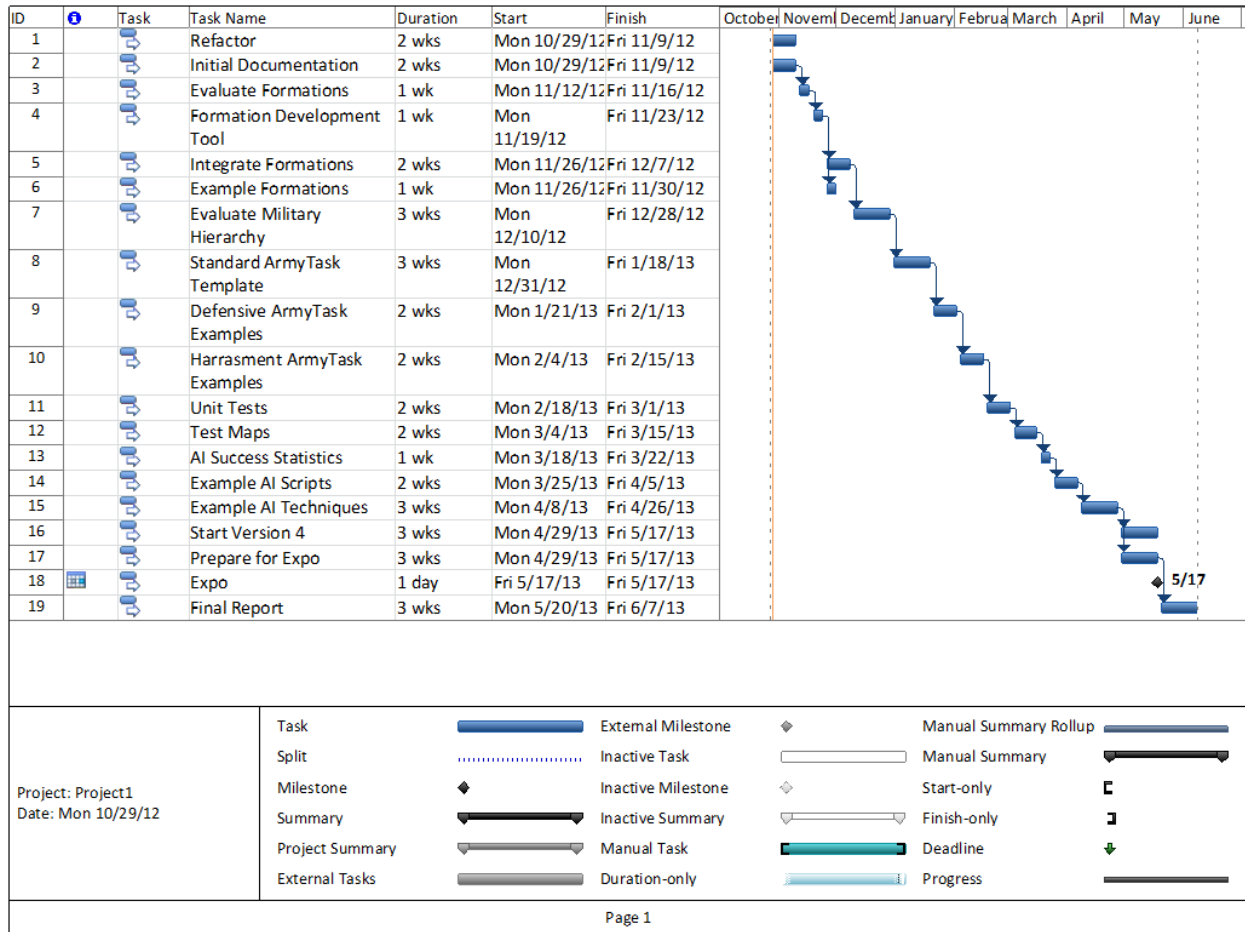
One of the things that could go wrong with this project would be if we fail to document clearly or thoroughly.

#### **Testing:**

Our testing method will consist of running mach games against the built-in AI on standard and custom maps and logs results.



## Preliminary Timetable:



## Roles of the different team members:

Jonah will be the primary programmer. Savannah will be the dedicated documenter. One of our end goals will be to have a manual for the framework we create.

## Integration Plan:

The way we have divided the project we should be able to work on both aspect simultaneously. The plan is to have Savannah document everything as it is being programmed by Jonah.

**References:**

Brood War API site: <http://code.google.com/p/bwapi/>

**Glossary:****ArmyTask:**

Each ArmyTask is set of behaviors used to complete certain tasks such as attacking or defending a specific location.

**Brood War API:**

An API designed to allow programmers to create AI for StarCraft and its expansion Brood War.

**FireTeam:**

A homogenous group of military units (An example: Terran's marines).

**Maps:**

Different maps are the setting where games take place.

**Modular:**

Dividing up the code into logically separated components so no one component depends on another.

**Protoss:**

One of the three character races in StarCraft that can be chosen. Protoss is a race of humanoids with advanced abilities.

**StarCraft:**

StarCraft is a real-time strategy game that was released on March 31, 1998.

**Terran:**

One of the three character races in StarCraft that can be chosen. Terran is a race of exiled humans.

**Zerg:**

One of the three character races in StarCraft that can be chosen. Zerg is a race of insectoids.

**Final Requirements:**

Throughout the course of the project our focus shifted more toward providing stable tools for future users, and away from providing a complete AI player. Initially we had hoped to be able to complete the framework in time to work on more complicated example players, as well as add extra functionality such as unit formations and customized test maps. Ultimately, however, our client felt that our time would be better spent polishing the production module, military module, and documentation, rather than working on additional features.

## **Weekly Reports – P3's:**

**team25 CS 461/462/463 week04 P3 (team25)**

### **Starcraft AI Player**

#### **Progress**

We have most of the requirements document done, but we are missing a couple of the categories. We have talked to Alan Fern last Tuesday and will be filling in the missing fields this week. We have most of it already generally planned out, but we just need to determine what exactly we want to accomplish in the document.

#### **Problems**

The problems we are facing is the magnitude of the project. There are so many things that we can work on and optimize. We need to narrow down and distinguish between what we would like to get done and what is actually feasible.

#### **Plans**

What we will be working on this week is prioritizing our tasks and finalizing the requirements document. If we have time we will start going through the existing code to clean it up, get me (Savannah) up to speed with everything that has been done, and start clearly commenting the code.

**team25 CS 461/462/463 week05 P3 (team25)**

### **Starcraft AI Player**

#### **Progress**

We just finished the requirements document. We're having some trouble with getting the Gantt Chart in the document. But we are ready to start with version 1.

#### **Problems**

We are using the Citrix Receiver provided by OSU to access Microsoft Project, but we were experiencing issues with trying to get the overall chart into the Word Document. We emailed Mike Bailey for help.

#### **Plans**

This next week we will be starting on version 1 which is the cleaning and documenting of the existing code.

### **team25 CS 461/462/463 week06 P3 (team25)**

#### **Starcraft AI Player**

##### **Progress**

This week we were able to get my (Savannah's) laptop set up to work with the code. I already had the 2010 Visual Studios installed so then we just installed the BWAPI, Visual Studios 2008, and Windows version control software. I also purchased StarCraft so I can play and learn what I need to know about the game in order to succeed with this project. We also ran through more of the code to help me get more familiar with what has been done with the code already. We were also discussing ways we want the documentation manual to be formatted.

##### **Problems**

We ran into problems when we were trying to get my computer to pull the code from the repository. We have been mostly just working on Jonah's laptop up until now and we didn't think about it until today. He has emailed Alan Fern about getting me access to the code.

##### **Plans**

This next week we will be working on refactoring the code and doing more documentation.

### **team25 CS 461/462/463 week07 P3 (team25)**

#### **Starcraft AI Player**

##### **Progress**

(Written by Savannah) So this week we started out with trying to get me access to the code, but unfortunately I need an EECS account. I have talked to Colisse Franklin and I will be meeting her on Tuesday to have it created. Jonah and I have played a bit more rounds of StarCraft so I can get a better feel about how the game works. We then talked a lot and looked through the code and tried to decide how we want to organize and refactor things to make it modular. We then drew up the outline of how we would like the manual to look and what needed to be done next.

## **Problems**

Our original plan was to have all the refactoring done by this week, but it was a much bigger task than we anticipated. In order to make it as modular as we would like it to be it will take a lot more time.

## **Plans**

Now that we have figured out how we would like the code to work we can now start refactoring and documenting.

**team25 CS 461/462/463 week08 P3 (team25)**

### **Starcraft AI Player**

## **Progress**

This week we created the poster outline. All we need to do is fill in the details which can be done really quickly. We started working on our elevator talks, but they still need a little bit of work. Savannah was able to get her EECS account created and successfully pulled the repository onto her laptop. We also made more notes about what we would like to accomplish and how we would like to refactor the code.

## **Problems**

We haven't made any solid changes yet because we were having trouble figuring out how to fork from the original repository. We have emailed Alan Fern for directions.

## **Plans**

This week we will finish our posters and our elevator talks. We will hopefully hear back from Alan Fern and figure out how to fork on the EECS servers. Once we can successfully fork we can implement all of our plans that we've made for the code.

**team25 CS 461/462/463 week10 P3 (team25)**

### **Starcraft AI Player**

## **Progress**

This week we were able to make our own repository and start making changes. We submitted our poster outlines on Monday, and presented our elevator talks on Tuesday.

## **Problems**

We were having trouble forking due to lack of privileges on the schools servers, but we were able to get something working.

## **Plans**

Our plan is to continue to meet every Tuesday, Thursday, and Sunday throughout the break to catch up and get ahead as much as we can.

**team25 CS 461/462/463 week12 P3 (team25)**

## **Starcraft AI Player**

## **Progress**

We've made more plans for the other two adapters (Production and scouting). We scheduled a time to meet with Joe (he also worked on the original project last summer) next week on Thursday. We wrote and submitted the significance paper and made the block diagram on how the data is structured.

## **Problems**

We are meeting with Joe to talk about the code he wrote, and how we could change it so it can work with the adapters we would like to create.

## **Plans**

This coming week we will work more on the adapters and how we will then continue expanding the functionality.

**team25 CS 461/462/463 week13 P3 (team25)**

## **Starcraft AI Player**

## **Progress**

We met with Joe and figured out how everything he wrote worked together. We also figured out what does and does not need to talk to our High Manager. We are working on refactoring now. We also looked at the scouting manager and discussed different options on how to refactor that code. We have updated our status table, and we are continuing to document our code.

## **Problems**

The refactoring of the Production and Scouting managers may be a bit more difficult than the Army manager, but it shouldn't take too much more time.

## **Plans**

Next week we hope to have either the Production Manager done or the Scouting manager done along with more documentation.

**team25 CS 461/462/463 week14 P3 (team25)**

## **Starcraft AI Player**

### **Progress**

This week we were not able to successfully complete the Production module as we had hoped (but progress has been made), but we had made progress with the documentation of our code. Jonah figured out how to use Doxygen which is a great way for us to quickly generate documentation from source code into an HTML format. We were updating comments and figuring out how we can utilize this to make documenting the entire project clearly and concisely.

### **Problems**

We were running into some issues with the generation of the HTML formatted output. When Jonah ran it the "todo" list was not being formatted properly, but it did when I ran it. We compared Cygwin versions and noticed I had a newer version. Jonah updating the version fixed the issue.

### **Plans**

This week we will finish the Production Module, continue the documentation of code, and prepare for our presentation on the 12th.

**team25 CS 461/462/463 week15 P3 (team25)**

## **Starcraft AI Player**

### **Progress**

This week we were able to create the Production Module (the adapter between Joe's code and the High Manager). However, we are still working on integrating the adapter



into High Manager. We were also able to increase our documentation in the code itself which will help with the writing of the manual tremendously!

## **Problems**

We were running into some trouble figuring out some parts to Joe's code in order to integrate it into the High Manager. It shouldn't take much longer to get it working though.

## **Plans**

This next week we're going to take a closer look at the scouting module. We're going to figure out exactly how we would like to split up the existing code and make it modular.

**team25 CS 461/462/463 week16 P3 (team25)**

## **Starcraft AI Player**

### **Progress**

We were successfully able to get the Production Module integrated into High Manager. Except there's one portion that we're unsure about, and we have sent Joe an email to ask him for some clarification. We also talked about how we might want to start implementing the Scouting Module, but that is for next week.

### **Problems**

We ran into a couple issues while trying to test the code with the ChoasLauncher. Every time Jonah ran the game on his laptop he received an error, even when it was working on mine. With some research we found what was causing it. Hopefully it won't happen again.

### **Plans**

Next week is to wait to hear back from Joe, and start working on the Scouting Module.

**team25 CS 461/462/463 week17 P3 (team25)**

## **Starcraft AI Player**

### **Progress**

This week we updates several documents with more detailed documentation. We were able to finish the production order integration. We were also able to get started with the scouting module implementation.

### **Problems**

We just started with the scouting module. We not exactly sure how we want it to function, but we're working on it.

### **Plans**

This next week we will be working on the scouting module.

**team25 CS 461/462/463 week18 P3 (team25)**

### **Starcraft AI Player**

### **Progress**

We worked more on the documentation of a couple more files, and worked more towards getting the Scouting Module working. We also updated and submitted our Poster for the expo.

### **Problems**

The Scouting Module is very tricky. We knew this going in that trying to split it the way we want to will be difficult. So, splitting and getting it to function correctly is still a work in progress.

### **Plans**

This next we will hopefully have a better grasp on how we would like the Scouting Module to work and start working toward getting it fully implemented.

**team25 CS 461/462/463 week19 P3 (team25)**

### **Starcraft AI Player**

### **Progress**

We solidified the documentation on a couple more files, ended up cleaning up some code in those files, and looked more at the scouting module.

## **Problems**

We ended up focusing more on cleaning up and documenting code, and only looked at the scouting module a little bit. It is going to be really tricky and will take some time.

## **Plans**

Our plan for next week is the same as last week. More documentation and more work on the Scouting module.

**team25 CS 461/462/463 week20 P3 (team25)**

## **Starcraft AI Player**

## **Progress**

Unfortunately we were not able to make any progress this week due to preparation for finals.

## **Problems**

We talked some about the Scouting Module. We're still trying to figure out how we want to split that up.

## **Plans**

Next week we will continue with the project, and try to get the Scouting Module working.

**team25 CS 461/462/463 week21 P3 (team25)**

## **Starcraft AI Player**

## **Progress**

We have finished all of the modules. We are brainstorming how we would like the Expo to go, and what we need to accomplish to get there.

## **Problems**

There are many different things we would like to accomplish before the Expo, but we are limited on time. Deciding what to do will be difficult.

## **Plans**

We plan on writing up a few ideas on how we would like to present at the Expo and run those ideas by Alan Fern. By the end of this week we will hopefully have a solid plan and a start on the implementation.

**team25 CS 461/462/463 week22 P3 (team25)**

## **Starcraft AI Player**

### **Progress**

We continued documenting a couple more files in the doxygen format. We have contacted Alan Fern about what area he would like us to focus on for the Expo. And we have done some more brainstorming about what we would like to accomplish before the Expo. We will be meeting with Alan Fern sometime this week to solidify our plans.

### **Problems**

We're waiting for a convenient time to meet with Alan Fern to narrow down what he would like to see at the Expo. We already know that we are going to expand on the Army Module instead of the Scouting, but there's a lot of options within Army Module as well.

## **Plans**

We only have a couple more files to document, and once we get a chance to talk to Alan Fern we can start working on implementing more within Army Module.

**team25 CS 461/462/463 week23 P3 (team25)**

## **Starcraft AI Player**

### **Progress**

This week we talked to Alan Fern and we determined the direction in which we would like to continue. We will expand on our Army Module and get it ready for the Expo by finishing the Formations, creating new ArmyTasks, and write example AI scripts. We also received confirmation from Alan Fern so we can have a QR code on our poster linking to our DoxyGen Documentation.

### **Problems**

We don't know where we are going to host the code yet, but we will sort that out this week.

## **Plans**

The final poster is due at the beginning of May and we plan on getting the DoxyGen Documentation hosted somewhere and set up the QR code this week.

**team25 CS 461/462/463 week24 P3 (team25)**

## **Starcraft AI Player**

### **Progress**

This week we were working on getting a QR code set up for our poster. We were also working on creating simple AI scripts for demonstrations at the Expo. The scripts will also be incorporated into our Poster. We also were able to fix minor bugs in the code, and optimized several aspects of the code. We then added a new file with various convenience functions ( ability to refer to expansions by proximity to starting location).

### **Problems**

Trying to make the scripts we ran into a lot of issues with existing code. We were able to fix most of them, but they took quite a bit of time and there are some still left.

### **Plans**

We plan on finishing a couple scripts, and continuing with the optimization of the code. We hope to have a fairly successful AI implemented by the end of the week. We will also submit our final poster for the Expo.

**team25 CS 461/462/463 week25 P3 (team25)**

## **Starcraft AI Player**

### **Progress**

This week we submitted our final Poster. It includes a QR code that will link to our DoxyGen documentation for the duration of the Expo. We also continued the development of the sample scripts we will be using for the Expo.

### **Problems**

This week we were studying for finals along with with week which will make progressing for the scripts a little slow, but we will get them done!

## **Plans**

Work more towards getting the scripts and project ready for the Expo.

**team25 CS 461/462/463 week28 P3 (team25)**

## **Starcraft AI Player**

## **Progress**

We started getting the final report together. We have big plans for the project documentation on what needs to be installed to get it working, how to install all of the tools, and how to use the tools. We were also able to get all of the Terran units working so that any unit composition would be available.

## **Problems**

We are still in post Expo mode and haven't worked a great deal more on new things so we haven't really ran into any new problems. One of the major problems that we hope to work out before the end of the term is getting units to get out of each others' way. Right now units are able to be blocked by other units and get stuck. We would like to find a way to resolve this, but it has been something we've been working on for a while now.

## **Plans**

We plan on working more on trying to get the units to move out of each others' way, incorporate units' abilities, and continue our work on the final report.

## Poster:

# StarCraft AI Player

## “Teaching the world about AI”

Savannah Van Beek, [vanbeeks@onid.oregonstate.edu](mailto:vanbeeks@onid.oregonstate.edu)  
 Jonah Brooks, [brookjon@onid.oregonstate.edu](mailto:brookjon@onid.oregonstate.edu)  
 Client: Dr. Alan Fern, [aferr@eecs.oregonstate.edu](mailto:aferr@eecs.oregonstate.edu)

**OSU**

**CS Capstone Projects**

### Introduction

The goal of our project was to create an interface through which future students could quickly and efficiently program artificial intelligence for StarCraft.

Our framework encourages AI research by providing a solid and user friendly foundation.

### Conclusion

- Diverse feature set covering most aspects of the game
- Modular design for easy extension of the framework
- Extensive documentation to help newcomers quickly understand the code — [Linked here](#)



### Potential Uses

**Desired Action:**  
Create a strong economy by establishing a sequence of bases throughout the map

**Desired Action:**  
Organize and command a diverse military to control the flow of battle throughout the course of the match

**Desired Action:**  
Seek out enemy bases to evaluate their military standing and current unit composition

### Modules

**Production Module:**  
Maintains and expands the potential for both military and economic development

**Army Module:**  
Positions military units across the map in order to secure strategic locations, defend bases, and destroy enemy structures

**Scouting Module:**  
Gathers and stores information about enemy bases as well as unit placement and composition

### Results



**Results:**  
Above: Placing Buildings  
Below: Scouting Enemy Activity



## OUR FRAMEWORK

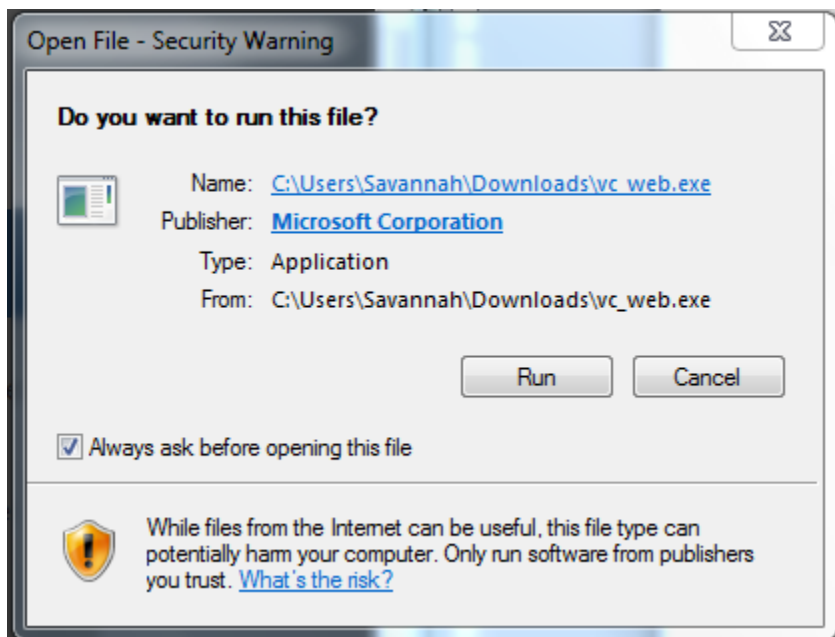
## B W A P I

05-17-2013

## Project Documentation:

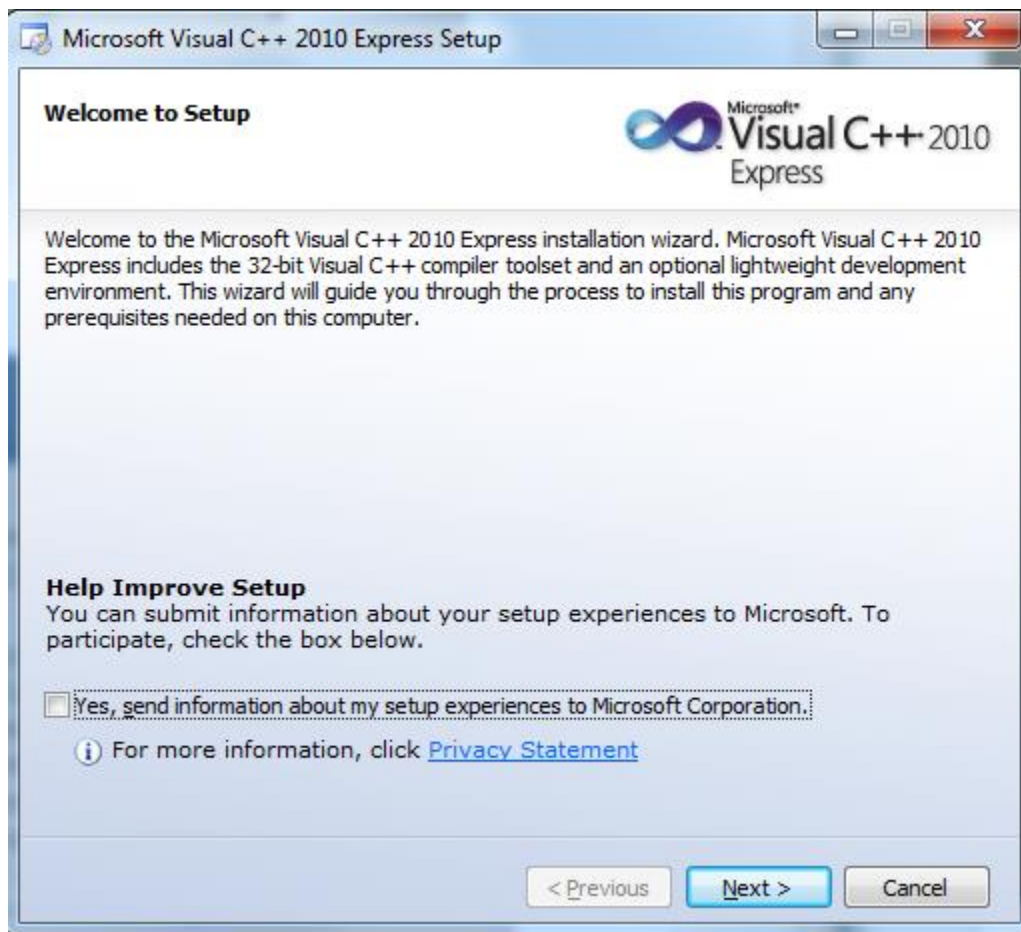
### Installing Visual Studios 2008 and 2010:

1. Install both Visual Studios 2008 and Visual Studios 2010. The code can only be compiled in Visual Studios 2008 and Visual Studios 2010 is used for convenience.
1. Download Visual Studios 2010:  
<http://www.microsoft.com/visualstudio/eng/downloads#d-2010-express> (This is a link for the express version, but the full suite works as well).
2. Run vc\_web.exe
3. Click Run

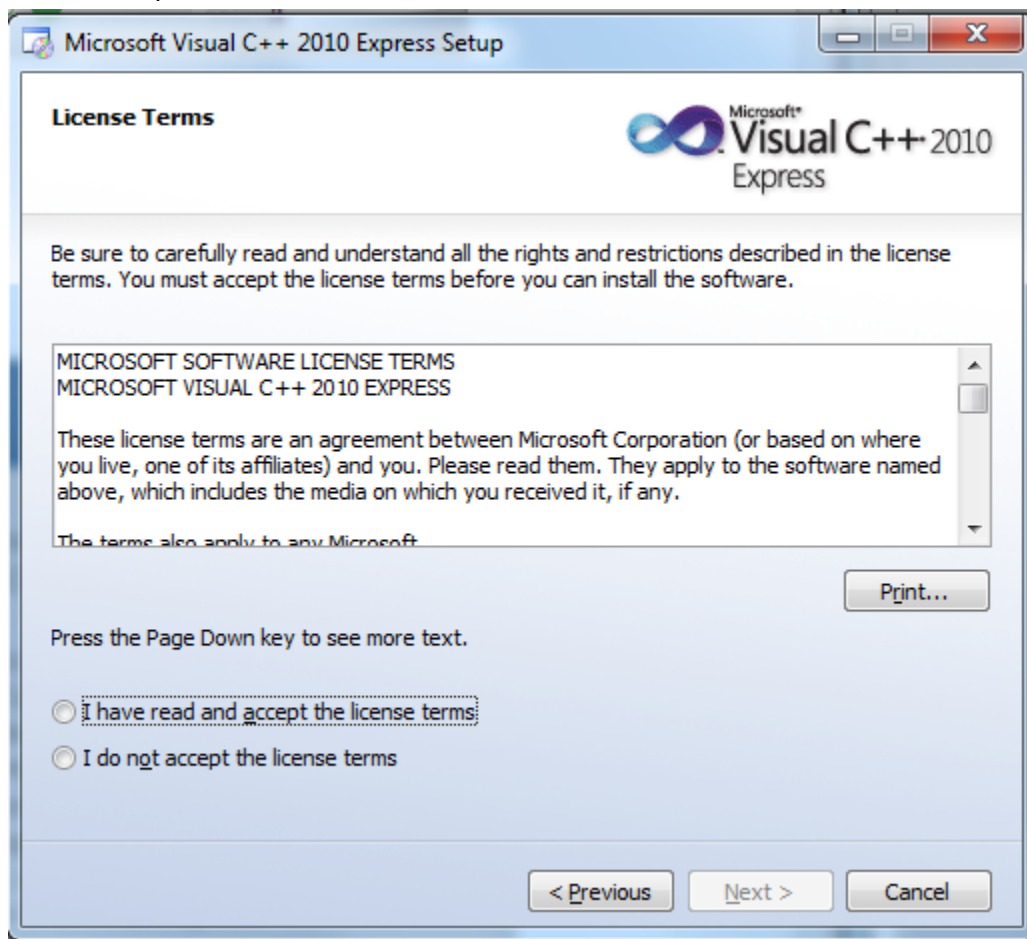




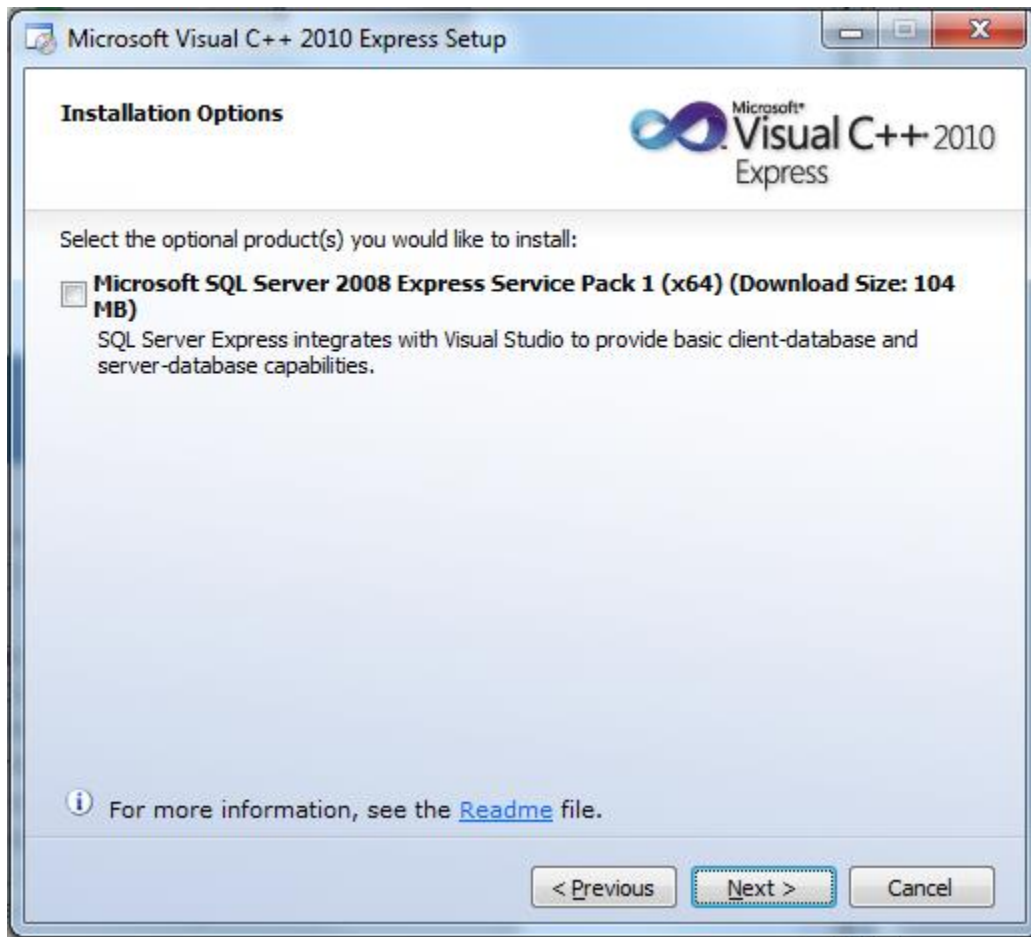
4. Uncheck box and click Next



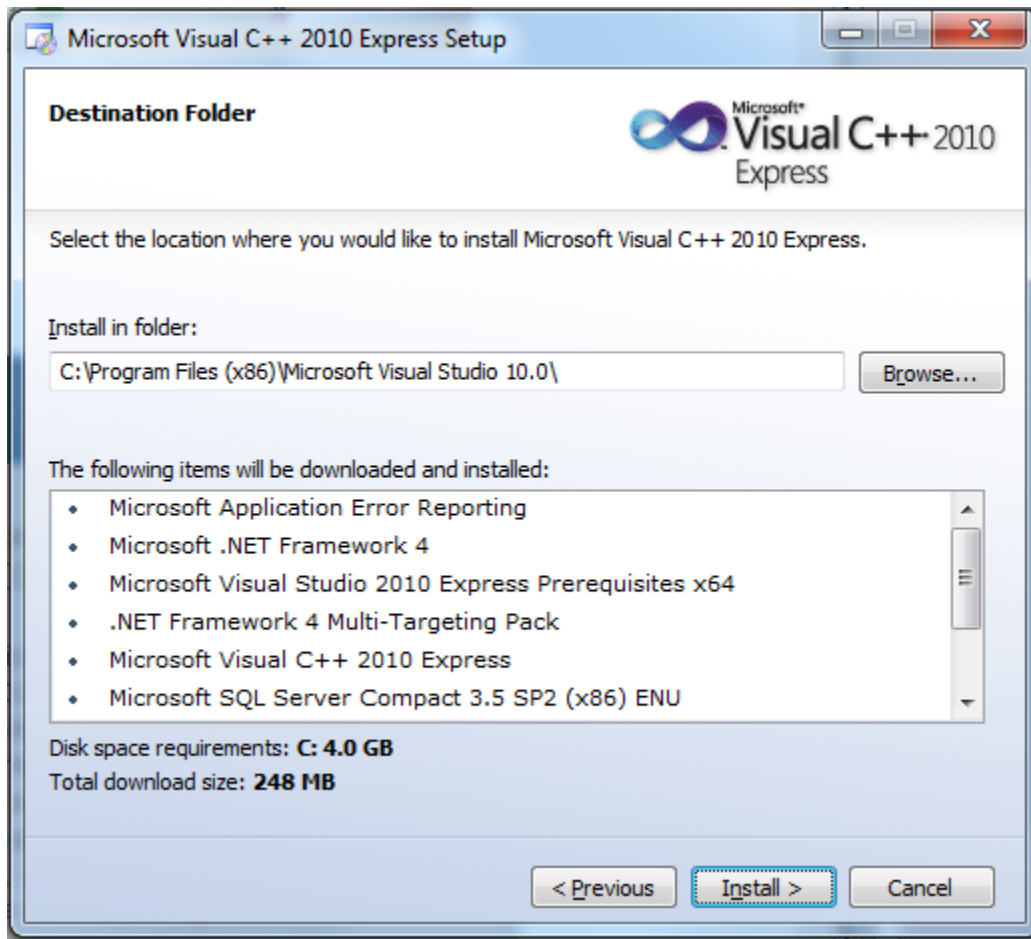
5. Check Accept and click Next



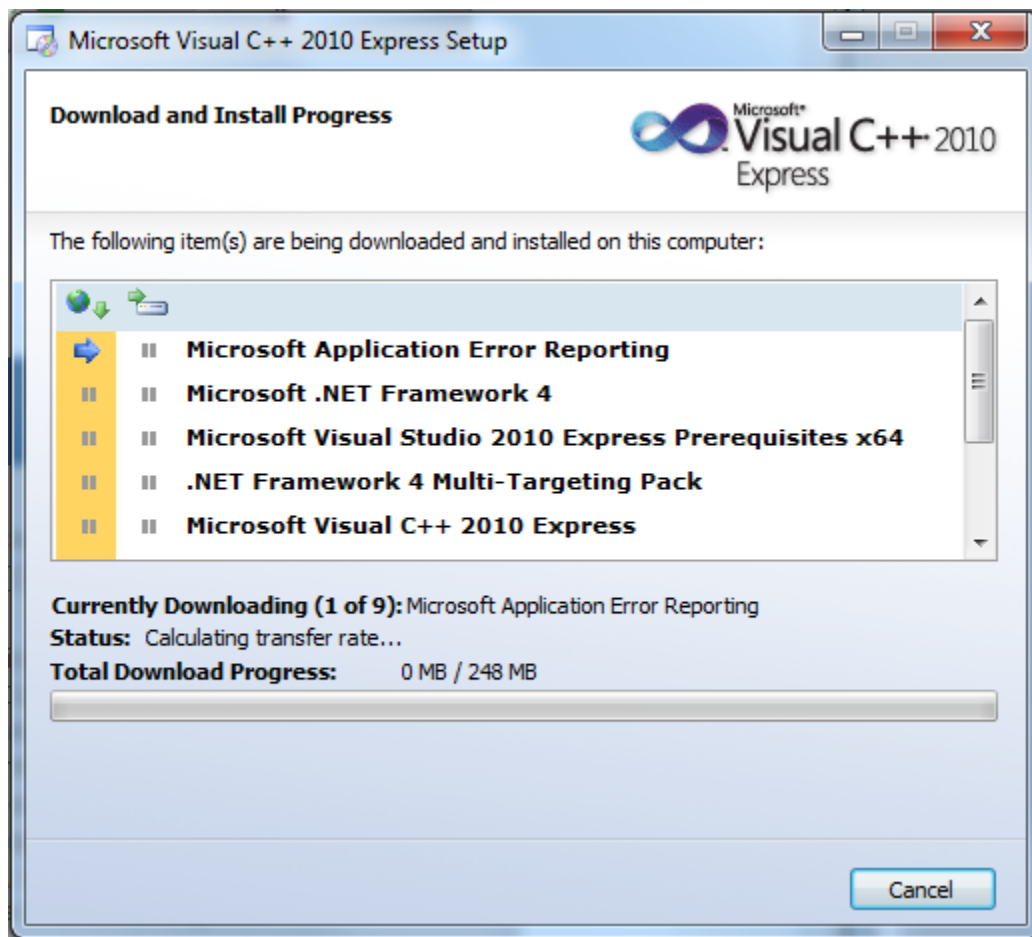
6. The service pack is not necessary, but can be installed without issues, click Next



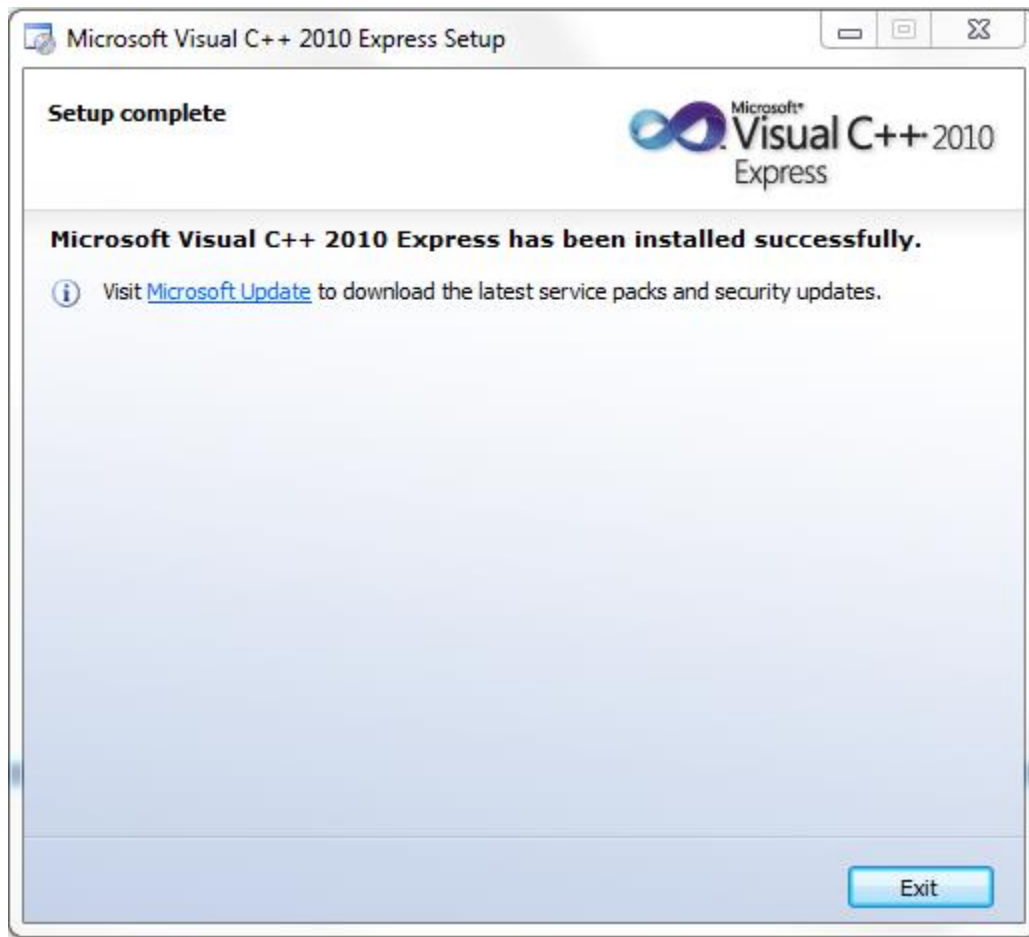
7. Change install directory if desired, then click Install



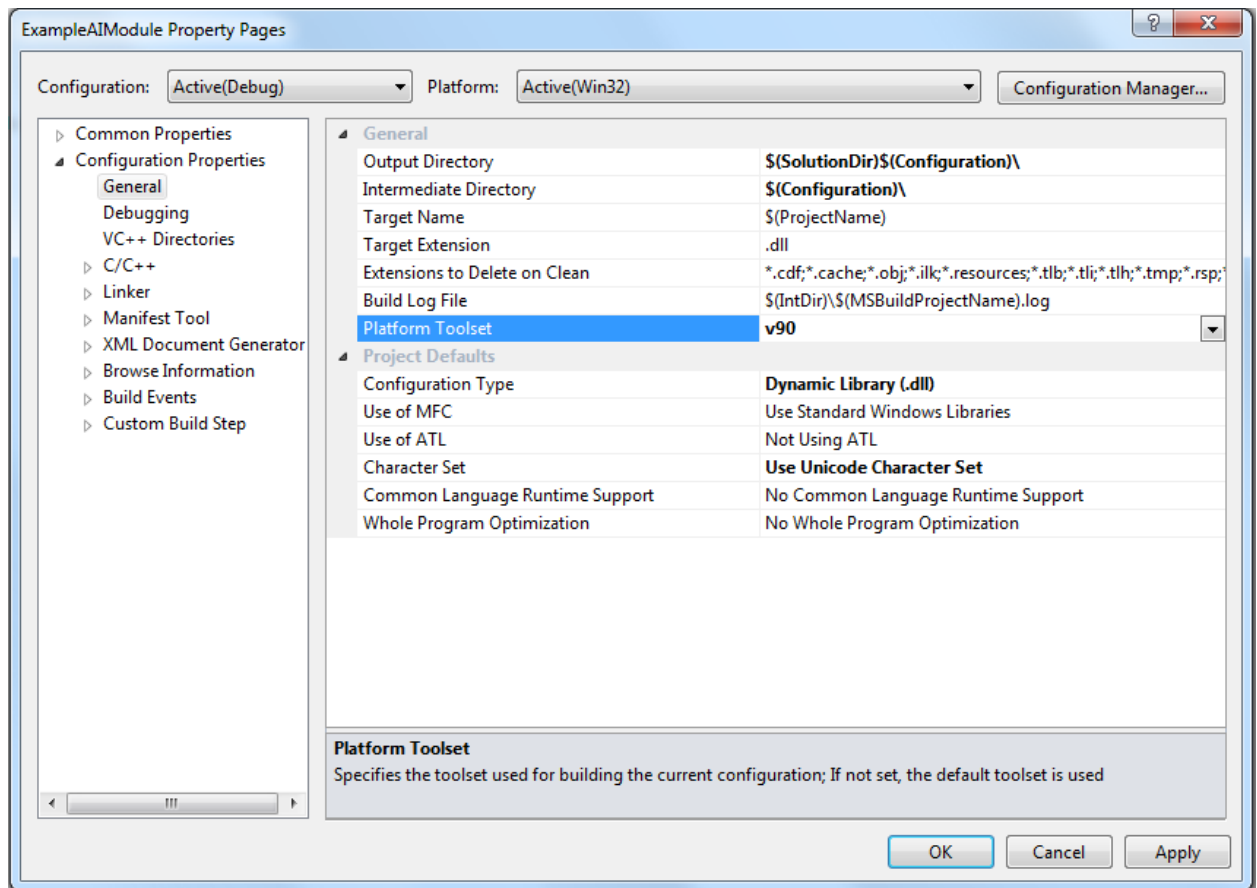
8. This will take some time..



9. Once it has completed click Exit

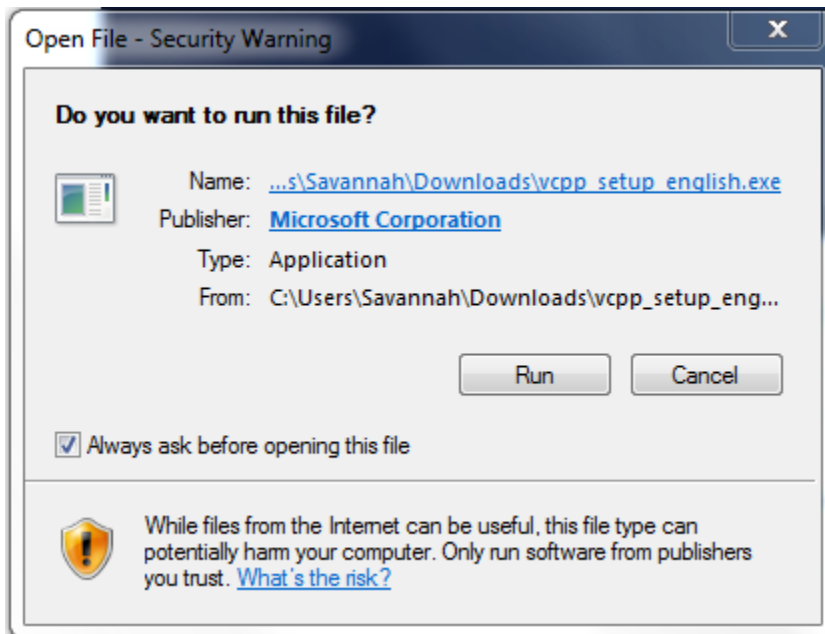


10. Open up Visual Studios 2010 and change the Platform tool set to be v90  
Right click ExampleAIModule -> Properties -> Configuration Properties -> General -  
>Platform toolset ->v90

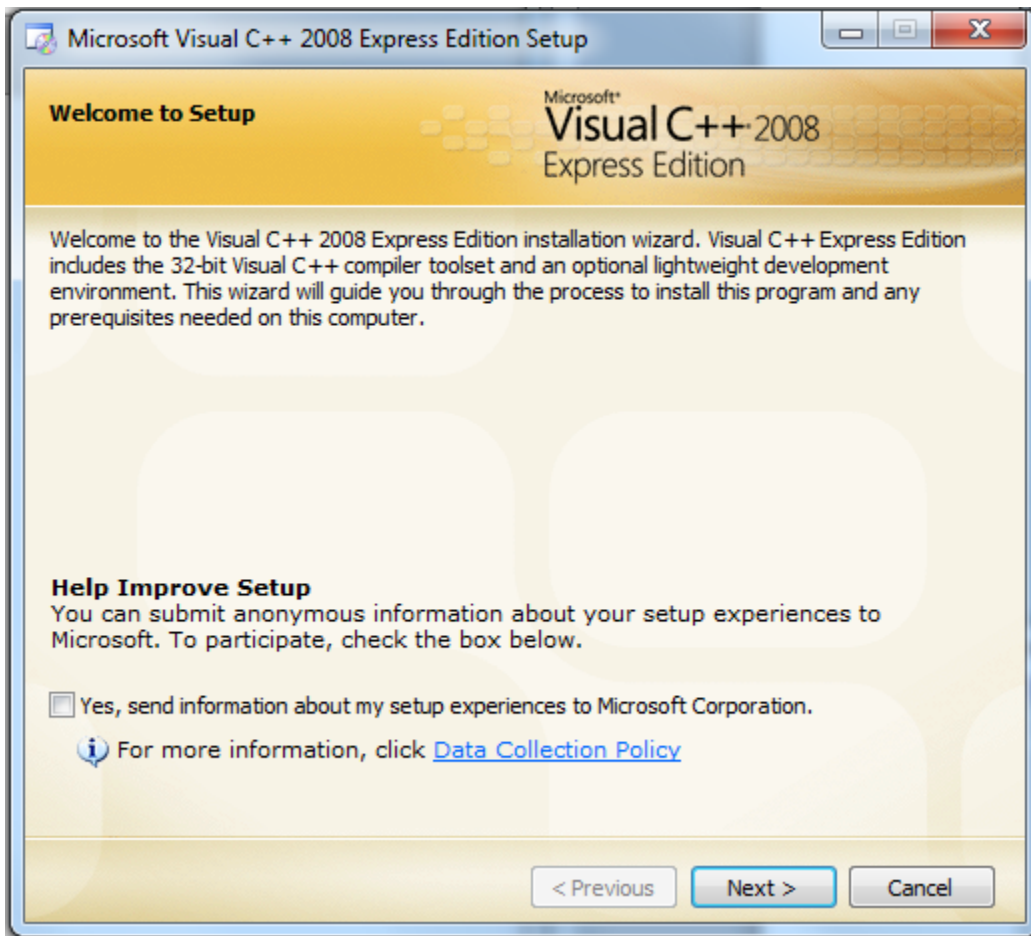


11. Install Visual Studios 2008. This will be slightly difficult since Microsoft seems to not have it listed on their website anymore, but it can be found here:  
<https://www.dreamspark.com/Product/Product.aspx?productId=34>

12. Click Run

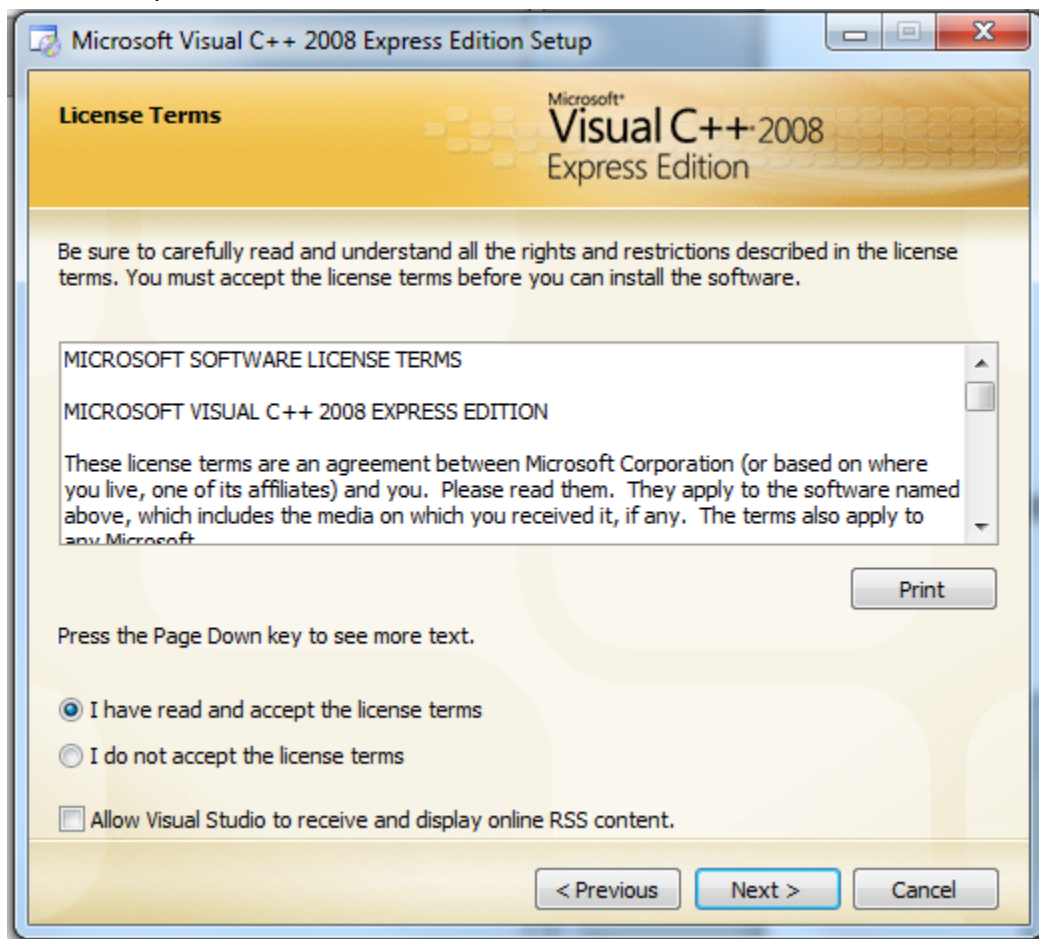


13. Uncheck box and click Next

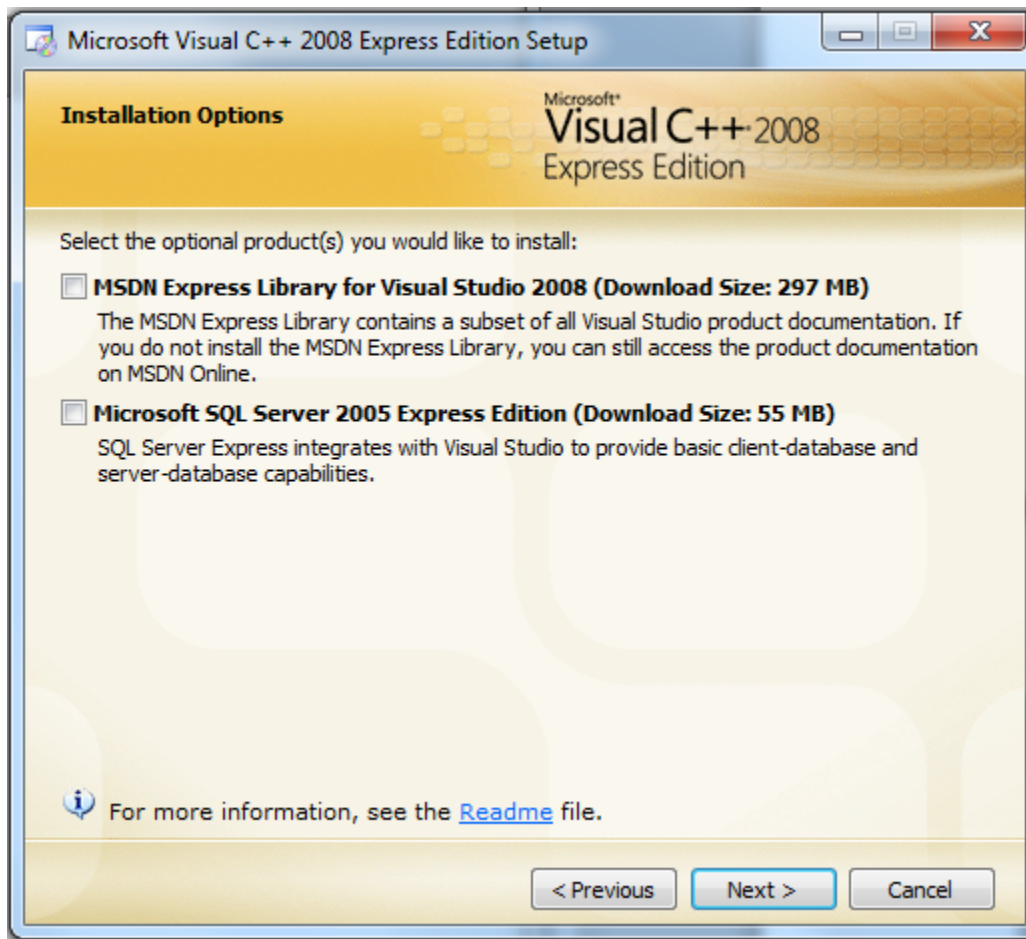




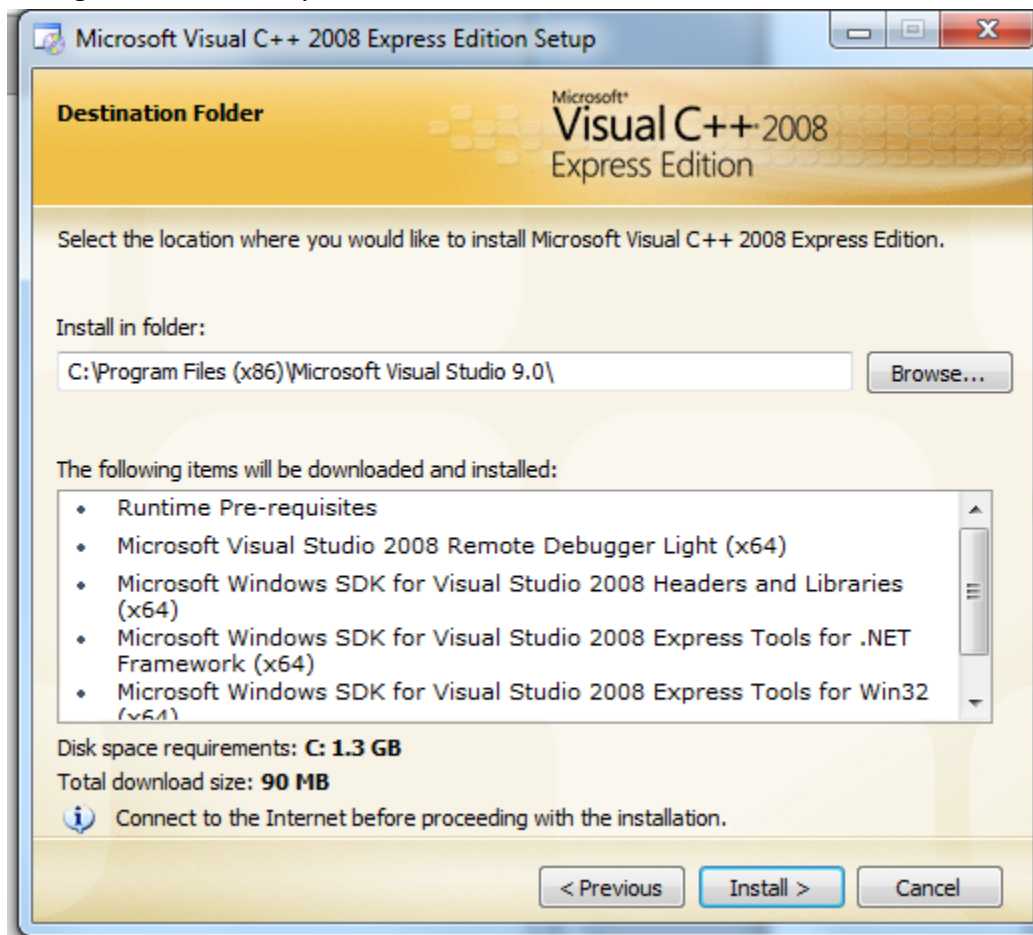
14. Check Accept then click Next



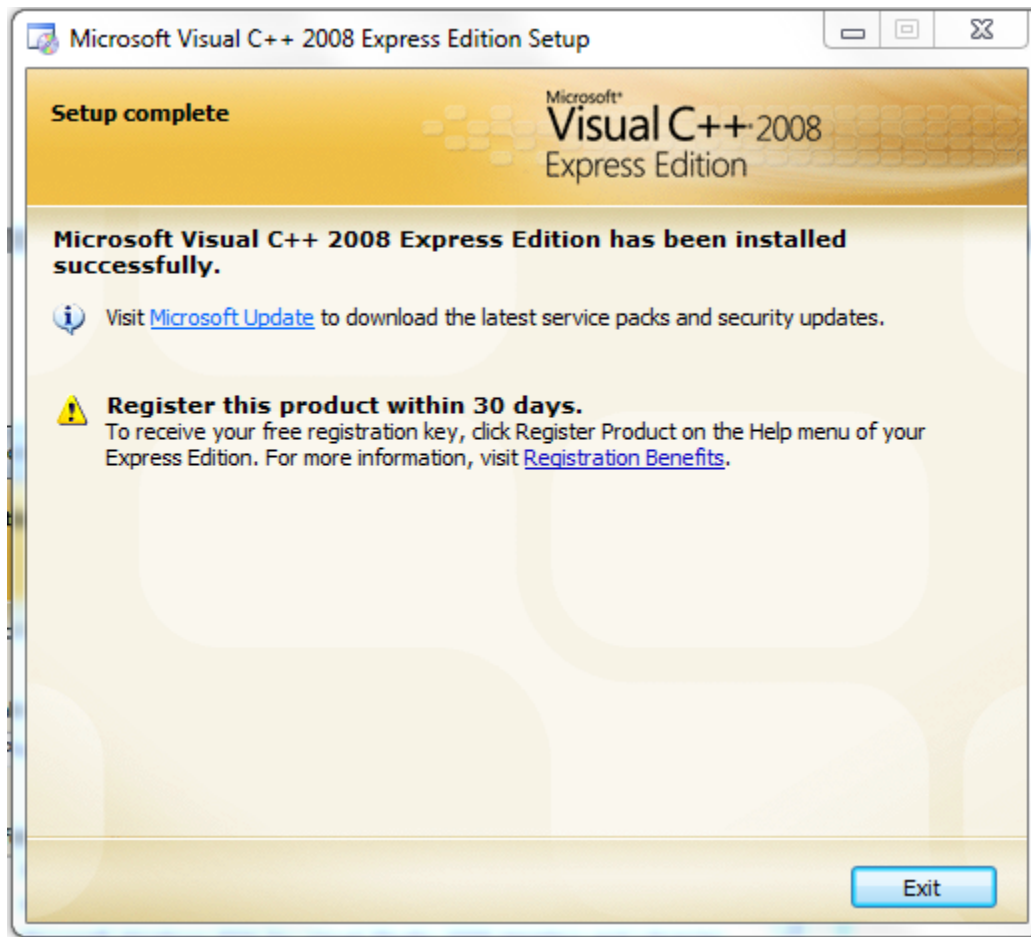
15. Both optional items are not necessary. Install if desired. Click Next



16. Change install directory if desired, then click Install



17. Registration may be necessary, click Exit




### Install StarCraft 1.16.1

1. <http://www.battle.net>
2. This will require a log in and purchase, but it is definitely worth it!
3. Download StarCraft – Brood War
4. Run it and patch it (it needs to be at version 1.16.1)
5. Start a single player game, create a character, then exit out
6. Start a multiplayer game and create a character
7. Instead of exiting start the game, uncheck “Show tips at startup”, then exit

### Install/ Configure BWAPI 3.7.3

1. Either follow the link below or search the site; just keep in mind that this version is depreciated and will only be seen under all downloads or specifically under the depreciated downloads

<https://code.google.com/p/bwapi/downloads/detail?name=BWAPI%203.7.3.7z&can=4&q=>

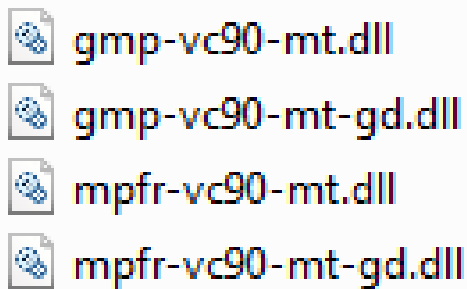
File:  **BWAPI 3.7.3.7z** 10.8 MB

Description: BWAPI 3.7.3 fixes several important bugs to get ready for tournaments in 2012. The BWAPI 3.7.x series will be used for this year's tournaments.

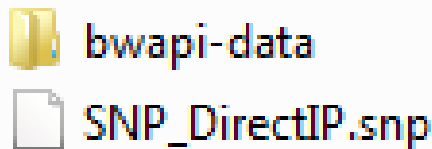
SHA1 Checksum: a470120fe372037a241e422046c814473a7b03b4 [What's this?](#)



2. Unzip BWAPI 3.7.3.7
3. Move the DLL files from (unzipped file)\WINDOWS\ to C:\WINDOWS\

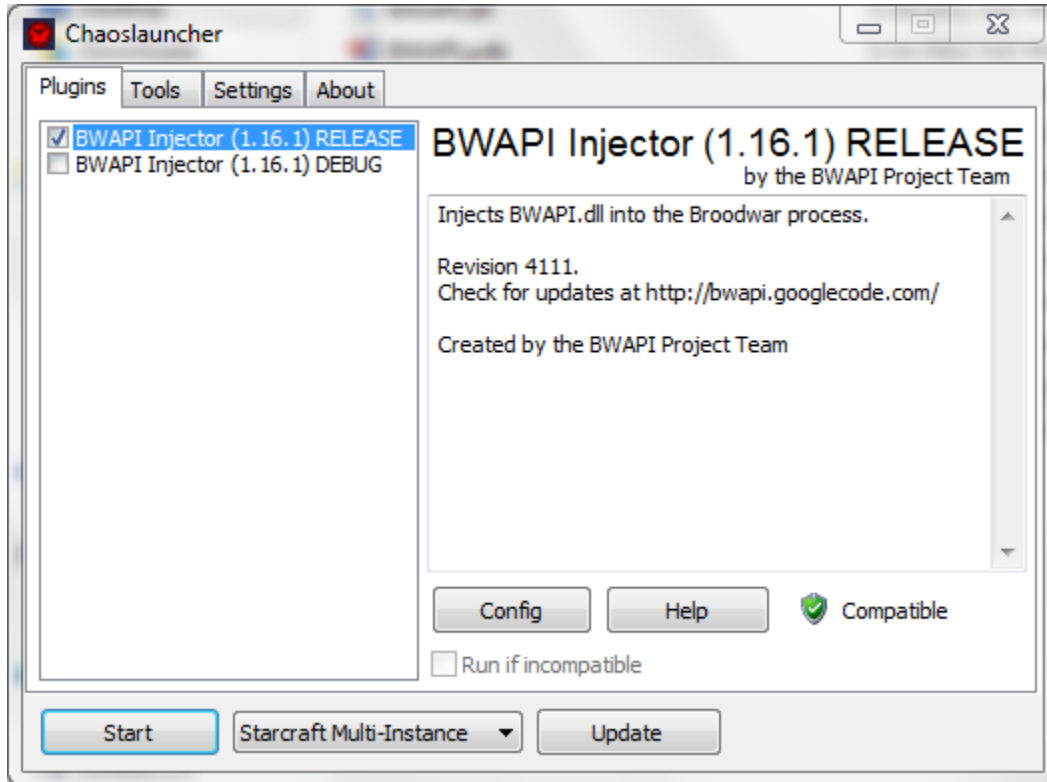


4. Move bwapi.ini from (Project directory)\other to (Unzipped file)\Starcraft\bwapi-data
5. Navigate to (Unzipped file)\Starcraft\bwapi-data and run Multiple Instance Hack.bat.  
This was developed by BWAPI in order to enable Multiple Instances
6. Move the bwapi-data folder and the SNP\_DirectIP.snp file from (unzipped file)\Starcraft\ to the StarCraft install directory, C:\Program Files\StarCraft by default

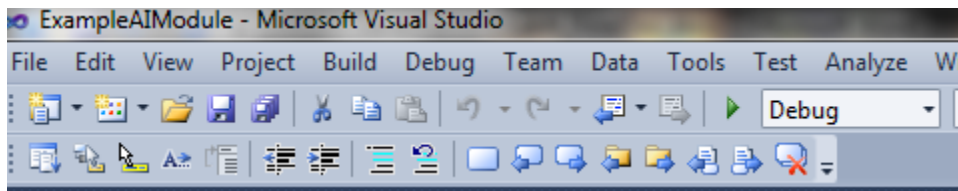


7. Copy BWAPI.dll and BWAPIId.dll from (Unzipped file)\Chaoslauncher\ to the StarCraft install directory, C:\Program Files\StarCraft\ by default
8. Navigate to C:\Program Files\StarCraft\Maps and create a folder named bwapi
9. Move the map from (Project directory)\other\2)Destination.scx to the newly created Maps\bwapi folder
10. Compile ExampleAIProject and copy the compiled ExampleAIProject.dll to the StarCraft\AI\ directory. Note that this can be done automatically by adding the following as a post-build event in VisualStudio: copy "\$(ProjectDir)Release\ExampleAIModule.dll" "C:\Program Files\StarCraft\bwapi-data\AI\ExampleAIModule.dll" /Y

11. Open Chaoslauncher-MultInstance.exe
12. From the drop down select Starcraft Multi-Instance
13. Make sure BWAPI Injector (1.16.1) RELEASE is selected
14. Click Start – This should then launch StarCraft and run the code

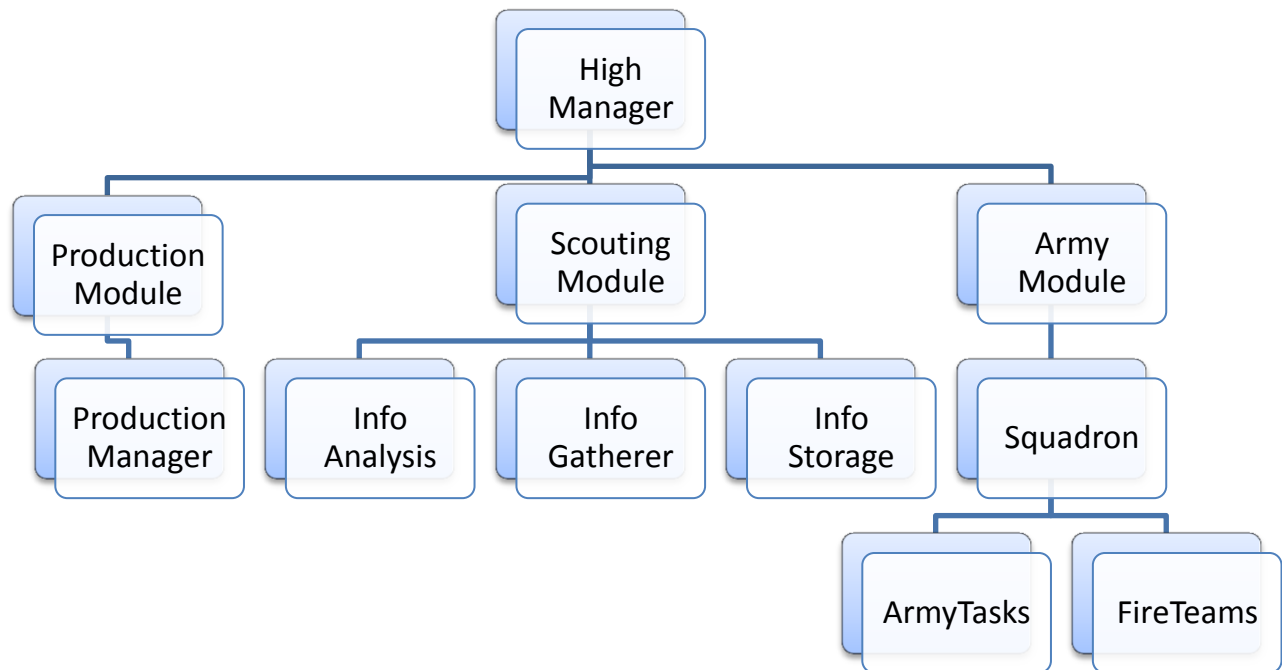


15. This can run in debug mode as well, but requires the following changes
16. The code needs to be compiled in debug mode



17. The BWAPI Injector (1.16.1) DEBUG must be checked
18. Then after hitting Start it will hang at the menu screen until Visual Studios is attached to the process via Tools ->Attach to Process ->StarCraft.exe (If StarCraft does not appear on the list, try checking "Show processes from all users")

## Block Diagram:



## Learned Technology:

1. <http://www.cplusplus.com>

Since this project was written entirely in C++, we found [cplusplus.com](http://www.cplusplus.com) to be an invaluable resource. This site contains a great deal of documentation and examples that proved very useful when working extensively with C++.

2. <https://code.google.com/p/bwapi/wiki/BWAPIManual>

This is the official site for the BWAPI. While the documentation for BWAPI is not anywhere near as thorough as [cplusplus.com](http://www.cplusplus.com), it was fairly useful for understanding how BWAPI functions.

3. <https://us.battle.net/support/en/article/starcraft-patch-information>

In the event that StarCraft has trouble connecting to the battle.net servers, this link allows direct download of the latest patch, which is required to run BWAPI.

## **What was Learned:**

**Savannah:** This year I learned many different things, mostly relating to programming and the C++ language. I already knew how to program in C++, but there were a lot of new aspects of the language that I was able to learn about while working on this project. I also learned about the process in which a project goes through while it is being developed. It was interesting to see how the requirements of the project and how some the focus can shift from one part to another.

**Jonah:** While most of the specific project details and requirements were provided by Dr. Fern, I had the opportunity to make a lot of decisions concerning project management and team structure. This gave me a chance to learn a great deal about how large projects are managed and completed. Having handpicked the members of my team, I learned the value of selecting compatible team members; this was by far the smoothest and least stressful project I have completed so far.



## Appendix A:

### Army Module.h

```
#pragma once
#include <map>
#include <string>
#include <BWAPI.h>
#include <BWTa.h>

#include <BWAM\Fireteams\Fireteam.h>
#include <BWAM\ArmyTasks\ArmyTask.h>
#include <BWAM\ArmyTasks\Specific\SpecificArmyTasks.h>
#include <BWAM\Squadron.h>

/** @file ArmyModule.h
 * @namespace AM
 */

namespace AM {
    /**
     * @typedef AID
     *
     * An identifier used to identify a Squadron.
     * Allows future users to change how armies are managed without breaking
     compatability with existing scripts.
     */
    typedef Squadron* AID;
}

/**
 * @class ArmyModule
 *
 * A singleton class used as an adapter to the military management files.
 */
class ArmyModule {

private:

    ArmyModule() {};
    static std::map<AM::AID, Squadron*> armyMap;    /**< Maps each army ID (AID) to its
    corresponding Squadron. */
```

public:

```
/**
 *   Initializes module prerequisites, such as each Specific Army Task.
 *
 *   @param none
 *   @return none
 */

static void init();

static void clearNonexistentUnits() { Squadron::clearNonexistentUnits(); }

/**
 *   Creates a new Squadron, returning the new AID as well as storing it in armyMap.
 *
 *   @see AM::AID
 *
 *   @param none
 *   @return an AM::AID identifying the newly created Squadron
 *   @todo    possibly add a deleteSquad function?
 */
static AM::AID createSquad();

/**
 *   Notifies the module that a unit needs to be removed.
 *
 *   @param unit The unit that is to be removed.
 *   @return none
 */
static void decommissionUnit(BWAPI::Unit*);

/**
 *   Transfers up to \a num units of type \a unitType from \a idFrom to \a idTo.
 *
 *   @param idFrom the AID of the Squadron from which units should be transfered.
 *   @param idTo the AID of the Squadron to which units should be transfered.
 *   @param unitType the type of unit that should be transfered between \a idFrom
and \a idTo.
 *   @param num the number of units that should be transfered between \a idFrom
and \a idTo.
 *   @return none
 */
```

```

static void transferTo(AM::AID idFrom, AM::AID idTo, BWAPI::UnitType unitType, int
num);

/**
 *   Transfers all units from \a idFrom to \a idTo, leaving \a idFrom's Squadron
empty.
 *
 *   @param idFrom the AID of the Squadron from which all units should be
transferred.
 *   @param idTo the AID of the Squadron to which these units should be transfered.
 *   @return none
 *   @todo What do we want to do with the empty Squadron?
 */
static void mergeWith(AM::AID idFrom, AM::AID idTo);

/**
 *   Adds \a unit to \a id's Squadron for automatic allocation.
 *
 *   @param id the AID of the Squadron to which \a unit should be added.
 *   @param unit the unit that is to be added to \a id's Squadron.
 *   @return none
 */
static void addUnit(AM::AID id, BWAPI::Unit* unit);

/**
 *   Returns the total number of units of type \a unitType accross all Fireteams
contained in \a id.
 *
 *   @param id the AID of the Squadron from which units of type \a unitType should
be counted.
 *   @param unitType the type of unit which should be counted.
 *   @return The number of units of type \a unitType accross all Fireteams contained
in \a id.
 */
static int getTotalOfType(AM::AID id, BWAPI::UnitType unitType);

/**
 *   Checks to see if \a id is currently in the process of completing its task.
 *
 *   @param id the AID of the Squadron whose task progress should be checked.
 *   @return True if \a id's current task is in progress, False if not.
 */
static bool taskInProgress(AM::AID id);

```

```

/**
 * Checks to see if \a id has completed its current task.
 *
 * @param id the AID of the Squadron whose task progress should be checked.
 * @return True if \a id's current task is completed, False if not.
 */
static bool taskComplete(AM::AID id);

/**
 * Checks to see if \a id has failed its current task.
 *
 * @param id the AID of the Squadron whose task progress should be checked.
 * @return True if \a id's current task is in a fail state, False if not.
 */
static bool taskFailed(AM::AID id);

/**
 * Returns the name of \a id's current task.
 *
 * @param id the AID of the Squadron whose task name is to be returned.
 * @return The name of the task \a id is currently assigned.
 */
static std::string getTaskName(AM::AID);

/**
 * Issues a new \a taskName task to \a id. Sets taskInProgress, taskComplete, and
taskFailed to false.
 *
 * @param id the AID of the Squadron which should be assigned a new task.
 * @param taskName the name of the task which should be assigned to \a id.
 * @return none
 */
static void task(AM::AID id, std::string taskName);

/**
 * Orders \a id to perform its previously assigned task, unless the task is already in
progress.
 *
 * @param id the AID of the Squadron whose task should now be performed.
 * @return none
 * @todo Return a bool indicating success (ie, false if current task is already
completed/failed/inProgress)
 */
static void doTask(AM::AID);

```

```

/**
 * Orders \a id to update the position of each of its units.
 *
 * @param id the AID of the Squadron containing the units that need to be moved.
 * @return none
 */
static void updateTeamMicro(AM::AID id);

/**
 * Instructs \a id to perform its future task(s) on or at position \a pos.
 *
 * @param id the AID of the Squadron whose task should be performed at position
\a pos.
 * @param pos the position at which \a id should eventually perform its task.
 * @return none
 * @todo Return a bool indicating success (ie, false if \a pos is invalid)
 */
static void setObjectivePosition(AM::AID, BWAPI::Position);

/**
 * Returns the position at which \a id has been ordered to complete its task.
 *
 * @param id the AID of the Squadron whose task position should be returned.
 * @return The position at which \a id has been ordered to complete its task.
 */
static BWAPI::Position getObjectivePosition(AM::AID id) { return armyMap[id]-
>getObjectivePosition(); }
};

```

## Production Module.h

```
#pragma once
```

```
////// TODO: Clean these up and standardize them. No quotes!
```

```
#include "ProductionOrder.h"
```

```
#include "ProductionDetails.h"
```

```
#include <BWPM\ProductionManager.h>
```

```
#include <BWTM\TaskManager.h>
```

```
#include <BWWM\WorkerManager.h>
```

```
class ProductionModule
```

```
{
```

```
private:
```

```
ProductionModule(void) { }  
~ProductionModule(void) { }
```

```
static ProductionManager *singleton;  
static ProductionOrder *order;
```

public:

```
/**  
 *   Initializes module's prerequisites.  
 *  
 *   @param none  
 *   @return none  
 */  
static void init();  
  
/**  
 *   Manages production tasks on each frame. Should be called from  
HighManager::onFrame().  
 *  
 *   @param none  
 *   @return none  
 */  
static void onFrame();  
  
/**  
 *   Handles various tasks upon the creation of a unit.  
 *  
 *   @param unit that has just been created  
 *   @return none  
 *   @see BWAPI::AIBModule::onUnitCreate()  
 */  
static void onUnitCreate(Unit* unit);  
  
/**  
 *   Handles various tasks upon the destruction of a unit.  
 *  
 *   @param unit that has just been destroyed  
 *   @return none  
 *   @see BWAPI::AIBModule::onUnitDestroy()  
 */  
static void onUnitDestroy(Unit* unit);  
  
/**
```

```

*      Handles various tasks upon the completion of a unit.
*
*      @param unit that has just been completed
*      @return none
*      @see BWAPI::AIModule::onUnitComplete()
*/
static void onUnitComplete(Unit* unit);

/**
*      Builds structures necessary for the construction of a specified \a unit (EX
UnitTypes::Terran_Marine).
*
*      @param unit whose dependencies should be resolved
*      @return none
*/
static void prepareTechFor(UnitType unit);

/**
*      Builds structures necessary for the construction of a specified \a upgrade (EX:
UpgradeTypes::Terran_Infantry_Weapons).
*
*      @param upgrade whose dependencies should be resolved
*      @return none
*/
static void prepareTechFor(UpgradeType upgrade);

/**
*      Builds structures necessary for the construction of a specified \a tech (EX:
TechTypes::Stim_Packs).
*
*      @param tech whose dependencies should be resolved
*      @return none
*/
static void prepareTechFor(TechType tech);

/**
*      Produces all currently available upgrades for the given \a unit type.
*
*      @param      unit type to be upgraded
*      @return none
*      @note Does not construct buildings required for upgrades; only produces
upgrades possible with current structures
*/
static void getAvailableUpgradesFor(UnitType unit);

```

```

/**
 * Produces a constant \a amount of units of given \a unit type.
 * If a unit of this type is destroyed then more are produce up to \a amount.
 *
 * @param unit type to be maintained
 * @param amount of units to be maintained
 * @return none
 */
static void maintainThisUnit(UnitType unit, int amount);

/**
 * Stops production module from actively maintaining the amount of \a unit.
 *
 * @param unit type to stop maintaining
 * @return none
 */
static void removeFromMaintenanceMap(UnitType unit);

/**
 * Expands to the base location nearest to \a tPos.
 * If ccNeeded is true this will also construct a new command center.
 * Otherwise it moves a non-active command center without building a new one.
 *
 * @param tPos Tile position near desired base location
 * @param ccNeeded whether or not a new command center should be built
 * @return none
 */
static void expandHere(TilePosition tPos, bool ccNeeded);

/**
 * Adds \a amount of \a unit type to the current production order.
 * The production module will then attempt to use ratios to maintain unit
composition.
 * Example: addToOrder(UnitTypes::Terran_Marine, 3);
 *           addToOrder(UnitTypes::Terran_Medic, 1);
 * should produce 3 marines for every medic while continually producing units.
 *
 * @param unit type to be produced
 * @param amount to be added
 * @return none
 */
static void addToOrder(UnitType unit, int amount);

```



```

/**
 *      Unimplemented
 *
 *      @param none
 *      @return none
 */
static void setAcceptableLosses(float minLosses, float gasLosses);

/**
 *      Unimplemented
 *
 *      @param none
 *      @return none
 */
static void resetAcceptableLosses();

/**
 *      Produces \a upgrade if currently possible.
 *      If prepareTech is true this calls prepareTechFor before requested upgrade.
 *
 *      @param upgrade to be researched
 *      @param prepareTech whether or not new buildings should be produced in order
to research given upgrade
 *      @return true if upgrade can be produced immediately otherwise false
 *      @todo If prepareTech is true this should research \a upgrade when
dependencies are met
 */
static bool researchUpgrade(UpgradeType upgrade, bool prepareTech = true);

/**
 *      Produces \a tech if currently possible.
 *      If prepareTech is true this calls prepareTechFor before requested tech.
 *
 *      @param tech to be researched
 *      @param prepareTech whether or not new buildings should be produced in order
to research given tech
 *      @return true if tech can be produced immediately otherwise false
 *      @todo If prepareTech is true this should research \a tech when dependencies
are met
 */
static bool researchTech(TechType tech, bool prepareTech = true);

/**
 *      Produces one unit of \a unit type if currently possible.

```

```

*
*   @param unit type to be produced
*   @return true if new unit can be produce immediately otherwise false
*/
static bool produceThis(UnitType unit);

/**
*   Returns a map of all units currently being maintained.
*   It stores everything that has been sent to maintainThisUnit().
*
*   @param none
*   @return current map mapping maintained units to maintained amount
*/
static map<UnitType,int> getMaintenanceMap();
};

```