Stratego Refactoring Project
Milestone 4

CSSE 375

Software Construction and Evolution

5/12/2016

Logan Erexson, Jason Lane
Jacob Knispel, Steve Trotta

Table of Contents

Introduction

This document describes our digital version of the strategy board game, Stratego. Stratego is a game in which two players control 40 pieces each that represent units of an army. The objective is to capture the enemy's flag or to put them into a position where the have no valid moves. All of the rules can be found here. Our program emulates this game while adding a few new features. The program is designed to work for Windows 7 and Windows 10. For the rest of the document, we will simply refer to our version of the original, physical board game as "Stratego".

This document compiles a number of documents and guides for users, developers, and maintainers of Stratego. This includes a User Manual that describes how the game can be played, guides for installing, configuring, and maintaining the game, specifications for the software requirements and for the software architecture and design, and an explanation of our test strategy.

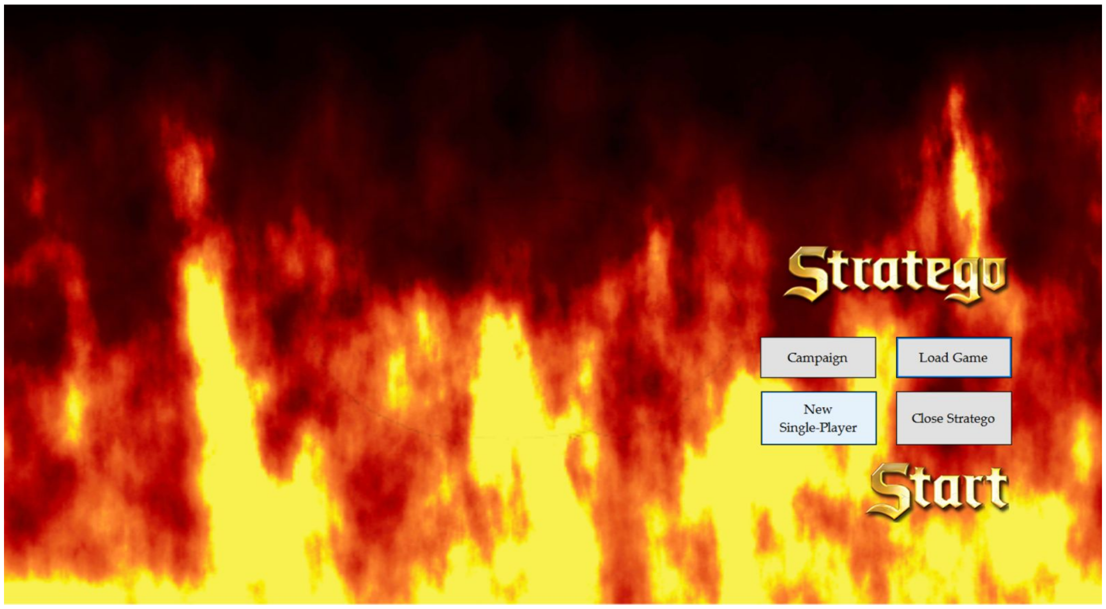
User Manual

Stratego features three distinct gametypes within the overall premise of the original rules:

- “Campaign” is a single-player gametype in which the human player is pitted against a simple AI opponent (the AI for all gametypes is currently in beta). As the player does not have a Flag piece in the campaign, they can only lose by being unable to move any pieces (typically by losing all pieces). The main distinct difference in Campaign mode is the variety of map environments and obstacles--contrasted with the other modes’ one, static, balanced map.
- “Singleplayer” is the traditional Stratego experience with original rules, 40 pieces for each player, the default board, and an AI opponent of similar quality to the Campaign’s.
- “Multiplayer” is the traditional Stratego experience with original rules, 40 pieces for each player, the default board, and a turn-based flow in which one player performs an action, there is an intermission screen to prevent cheating, and the next player’s turn commences once control has been passed off. In Multiplayer, the method of input is a single, shared keyboard and screen--it is not networked as of yet, though future iterations *may* support this option.

Main Menu:

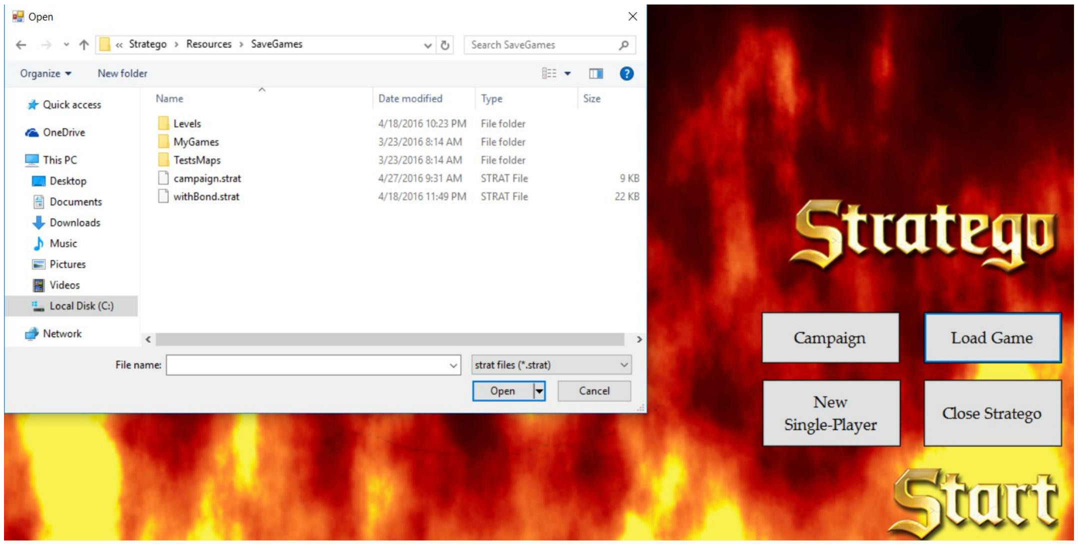
The Main Menu of Stratego is the launching point of all three of our gametypes.



To begin a new session of the Campaign gametype, simply click the Campaign button. Likewise for the Singleplayer gametype and the “Single-Player” button. The default gametype, Multiplayer, is initiated by pressing “Start”.

To load any previous game save, be it Campaign, Multiplayer, or Singleplayer, click the “Load Game” button in order to bring up this simple file navigator seen below

:



Name	Date modified	Type	Size
Levels	4/18/2016 10:23 PM	File folder	
MyGames	3/23/2016 8:14 AM	File folder	
TestsMaps	3/23/2016 8:14 AM	File folder	
campaign.strat	4/27/2016 9:31 AM	STRAT File	9 KB
withBond.strat	4/18/2016 11:49 PM	STRAT File	22 KB

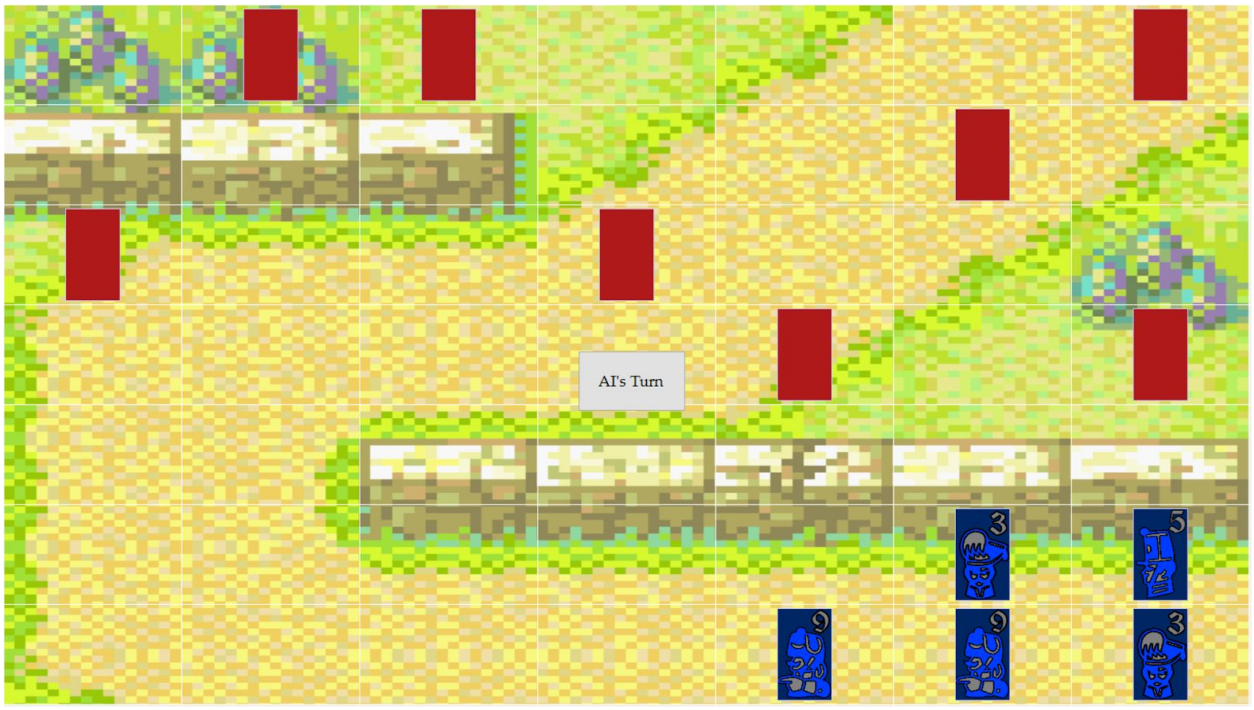
Navigate to the file you wish to load and double click the file or click “Open” in order to launch that game state and continue from where you left off.

The Konami code:

The Konami code alters the game, across game modes, in a few different ways when entered. The Konami code may be entered any time and stands as this combination of keystrokes: Up, Up, Down, Down, Left, Right, Left, Right. By default, it activations options for Movable Bombs, Movable Flags, use of nonstandard pieces, such as the BondTierSpy piece, (currently restricted to use in Multiplayer, may affect the Campaign in the future), and any battle-behavior changes pertaining to said pieces.
The effects of the Konami code are active until the user quits to the main menu screen, and can be re-activated at any point afterward.

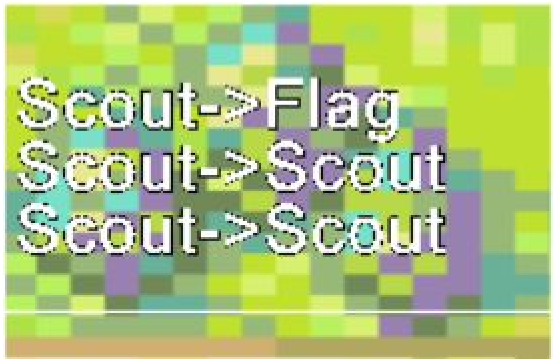
Campaign:

This is an example of a level of the Campaign. Notice the blue pieces at the bottom, signifying

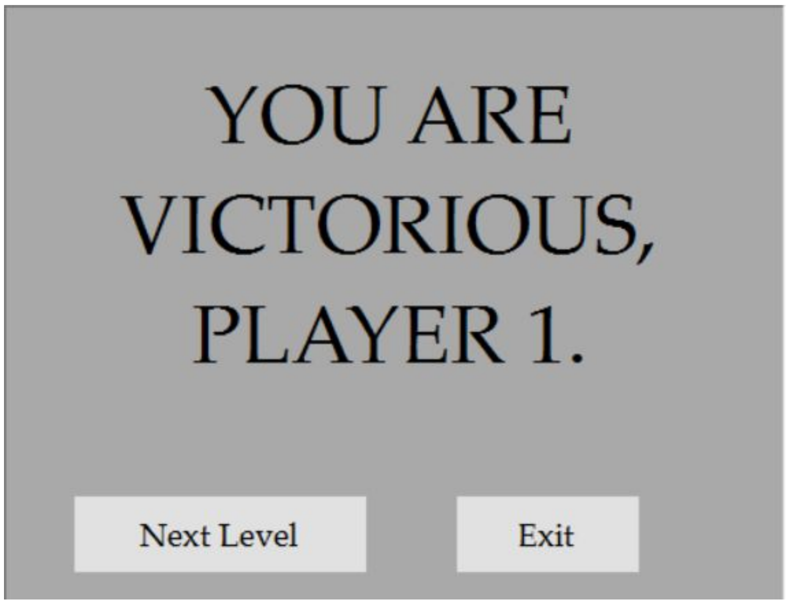


your standing army, and red pieces at the top signifying the enemy, notably the enemy flag, which it is the user's objective to capture. The turn-based combat is still the same as any other gametype, with control alternating with the AI and the player.

The Campaign shares the same Killfeed display as any other gametype. It shows the user the pieces that were captured in the order in which they were captured. These notifications will appear on the top left of the screen and will be of the format “Captain -> Soldier”, where Captain took the Soldier.

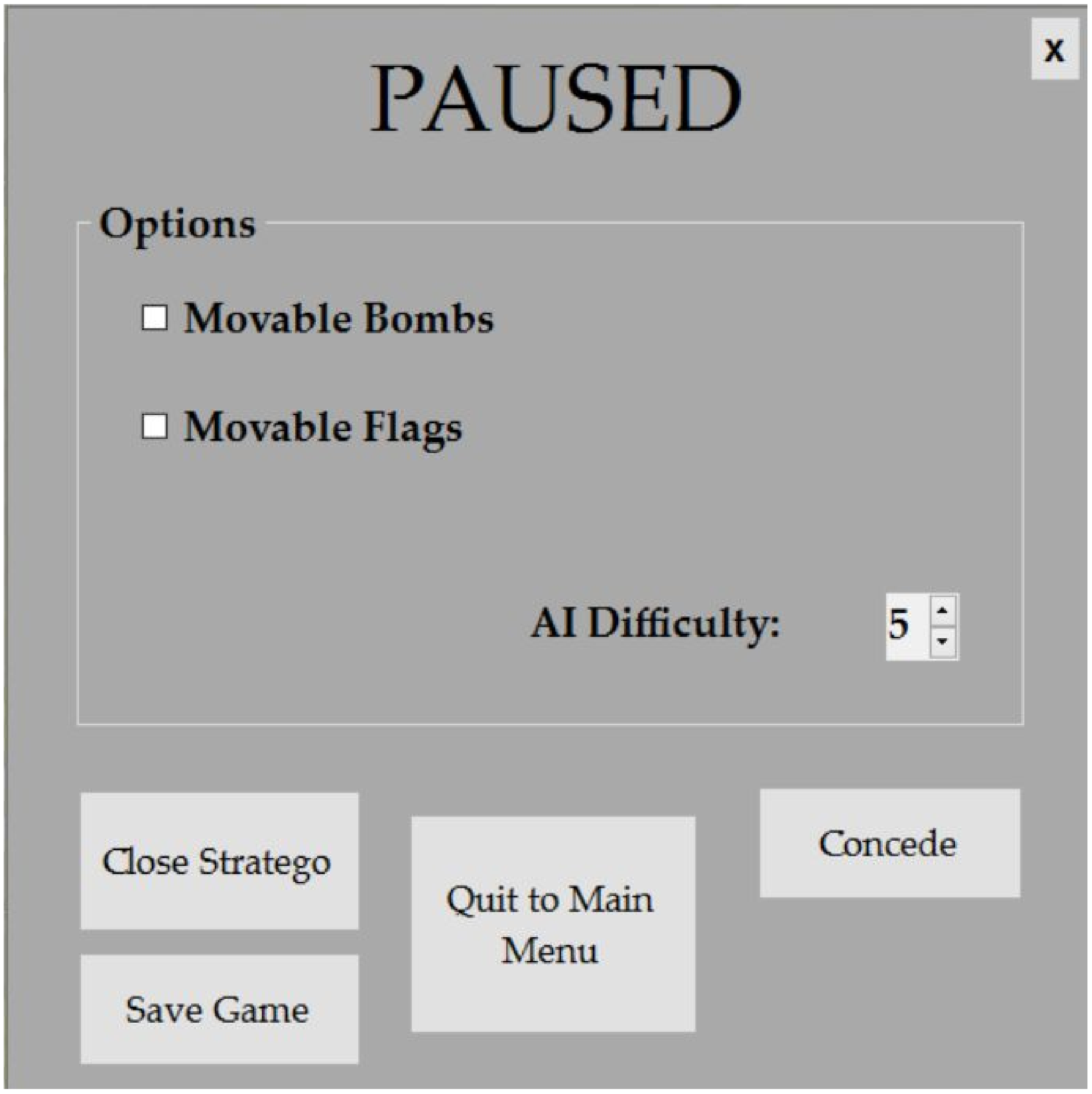

Scout->Flag
Scout->Scout
Scout->Scout

Once the flag is captured in single-player mode, a dialog box will appear, informing the player that the level has been beaten and giving the player a choice of exiting or continuing to the next level.


YOU ARE VICTORIOUS, PLAYER 1.

If the Konami code is active, a player may go to the next level without beating it by pressing the page up key in while the game is paused. They can also go back to previous levels using the page-down button. This allows the player to replay missions without completely restarting the session.

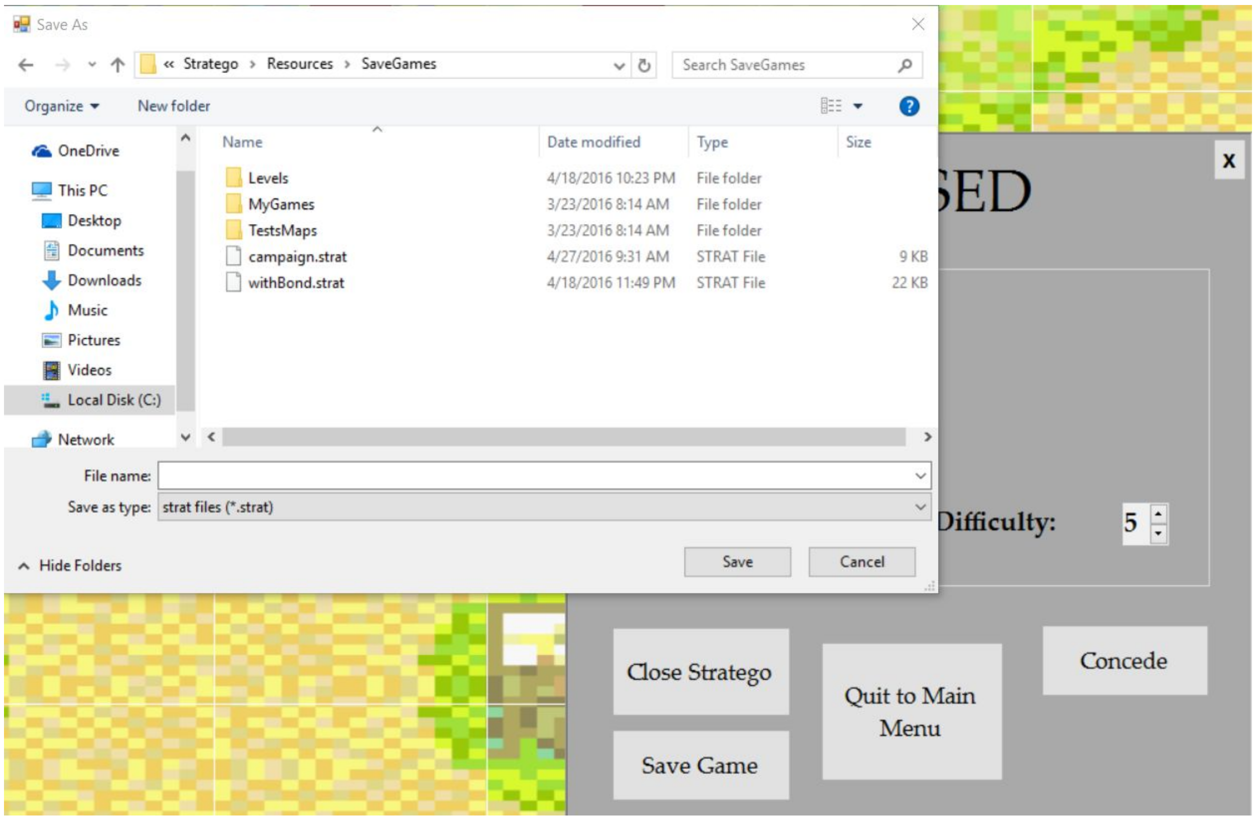
Pause Screen:



The pause screen can be accessed by pressing the Escape key while in game. It can be closed by pressing Escape again or by clicking the “X” button. You can access a number of actions while the screen is open.

The player may save the game-state (of any gametype) by selecting the “Save Game” button at the pause screen. The game may be saved in any location on the user’s machine using a typical file system navigator.

The game may be loaded again from the main menu by using the “Load Game” button to open the file system navigator again to go to and select the desired save file to load. When a save is loaded, everything is maintained as if you had made it that far in one sitting. Below is an example of saving the game:

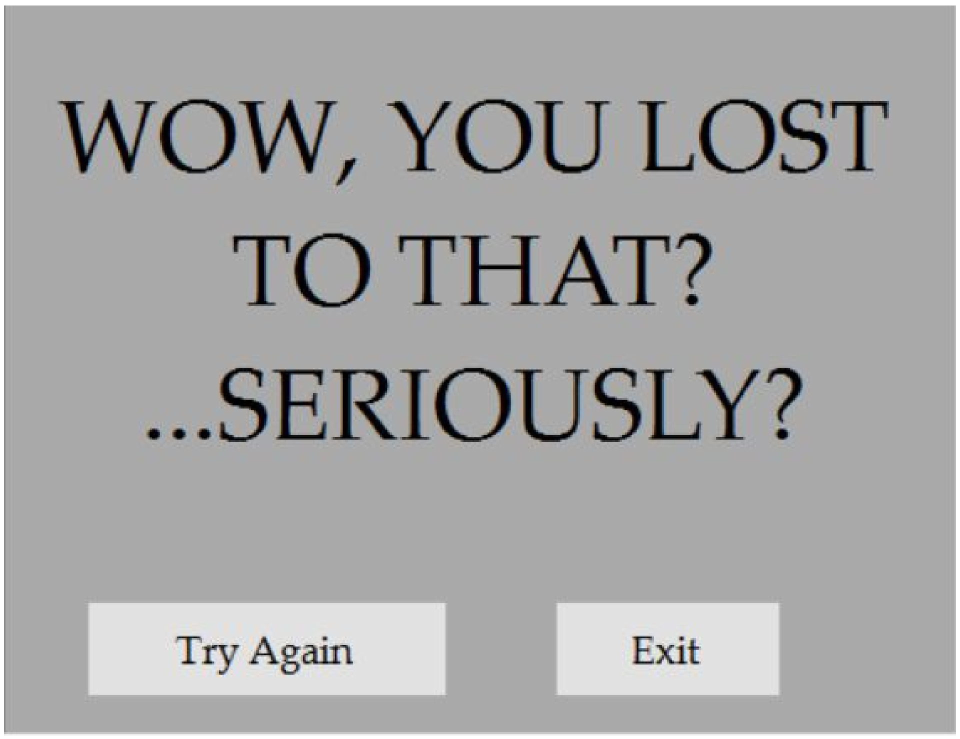


Name	Date modified	Type	Size
Levels	4/18/2016 10:23 PM	File folder	
MyGames	3/23/2016 8:14 AM	File folder	
TestsMaps	3/23/2016 8:14 AM	File folder	
campaign.strat	4/27/2016 9:31 AM	STRAT File	9 KB
withBond.strat	4/18/2016 11:49 PM	STRAT File	22 KB

File name:
Save as type: strat files (*.strat)

You may “Concede” from any gametype, to the Main Menu, by opening the pause menu and selecting the “Concede” button. This opens the “game lost” dialog that taunts the player and has options to restart the level - “Try Again” - or exit the game completely - “Exit”.

Losing the game (for the Campaign, this entails losing all of your pieces) will activate the “game lost” dialog as well, with the same options that pop-up gives the user.


WOW, YOU LOST TO THAT? ...SERIOUSLY?

The player can return to the main menu by opening the pause menu and selecting “Quit to Main Menu”.

The player can exit the application completely by opening the pause menu and selecting “Close Stratego”.

The player may choose the difficulty level of the opposing AI by opening the pause menu and incrementing the counter labeled “AI Difficulty” (higher is harder, lower is easier). At the current time, our AI is less than ideal, future iterations of the software would include this as a top-priority feature.

Single-Player / Multi-Player:

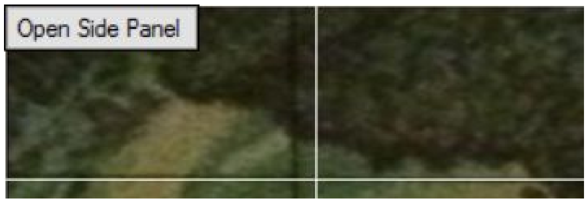
In single-player, the human player is pit against an AI player in the classic Stratego board and classic gameplay style. By “classic style”, we mean the board is a 10 x 10 grid with obstacle “ponds” in the middle and the ability for the AI and player to select their own pieces (with restrictions to the amount of certain pieces that can be placed).

In multi-player, the game functions exactly like the original board game with only one small twist. When turns commence, the current player (whose pieces are visible) selects his/her turn and plays. After the current player is finished, the screen will obscure all pieces on the board and wait for the acknowledgement that the next player will now assume control. This is in order to prevent the opposing player from seeing the current user’s setup when switching turns.

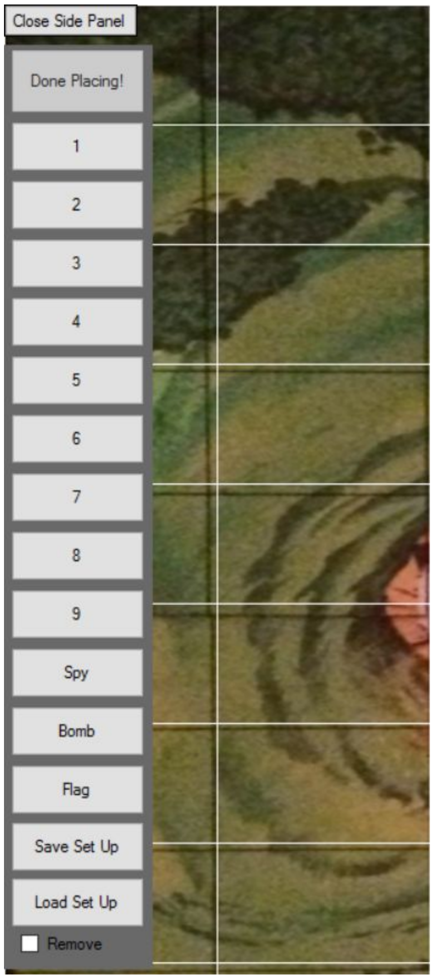
Other than the differences described in the above two paragraphs, Single-Player and Multi-Player content can be seen as identical. One simply has an AI opponent while the other a human one.

To select a human controlled piece type to place on the board, the player can either click the “open side pane” button at the top left of the screen in order to reveal a button panel of piece types, or use the keys of the keyboard directly. The key and button mapping is as-follows:

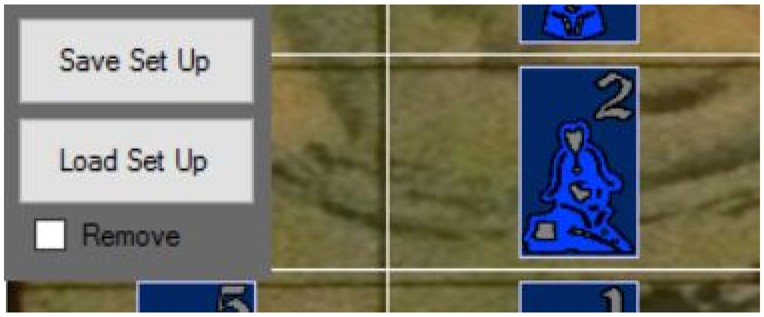
1 - Marshall
2 - General
3 - Colonel
4 - Major
5 - Captain
6 - Lieutenant
7 - Sergeant
8 - Soldier
9 - Scout
B[key] / Bomb[button] - Bomb
S[key] / Spy[button] - Spy
F[key] / Flag[button] - Flag



This is the side panel button - to open the piece button panel.

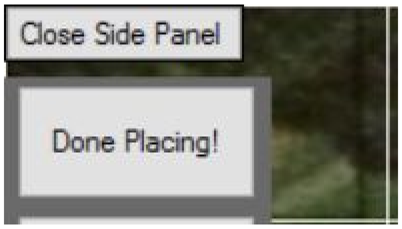


This is the side panel itself, you may select a piece to place, select the remove tool (to delete piece from the set-up), save the current board set-up, or load a board set-up.

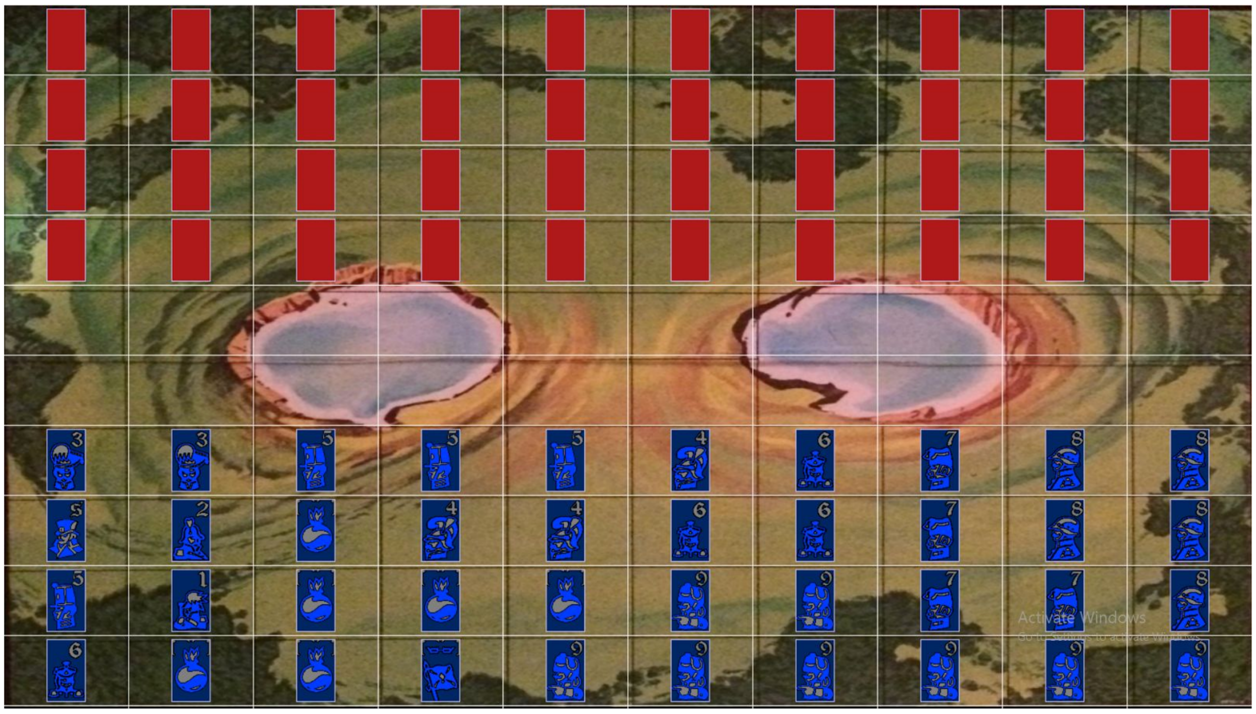


With the Save Set Up button, the file navigator will be opened and the user may save the current piece locations for his/her army.

Load Set Up opens the same navigator and places the pieces of the selected formation on the board for the current game.

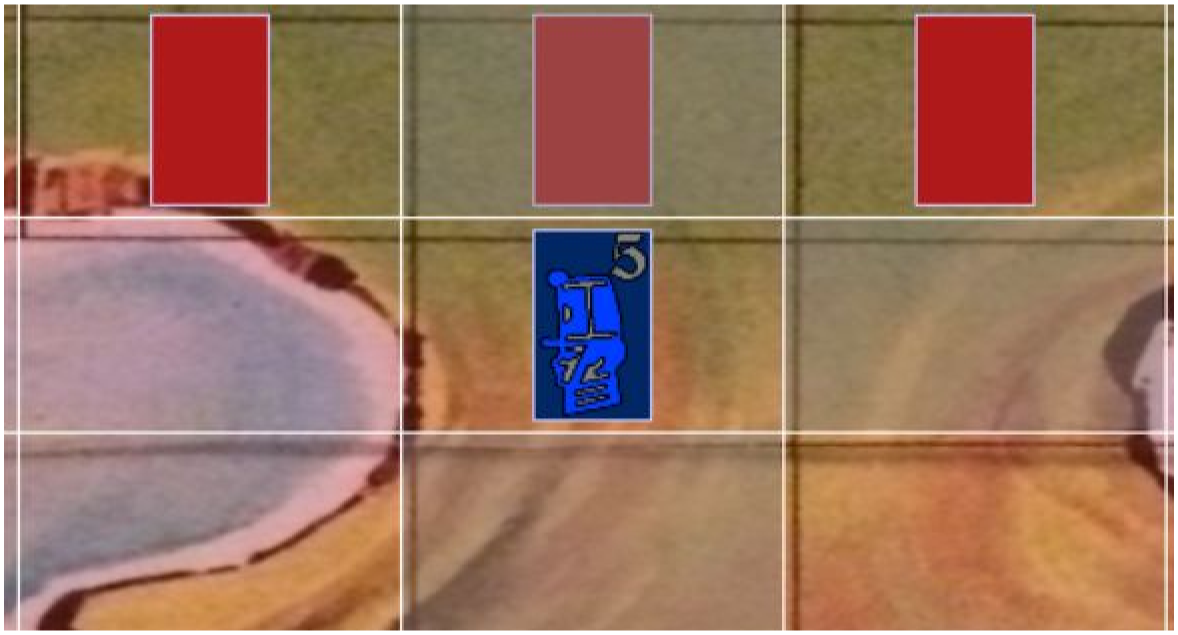


At the end of the piece placing phase, the player selects Done Placing! from the side panel in order to signal the next player's turn at placing pieces.



This is an example of a full board, ready to begin play.

Of final note is piece and turn selection. When one is trying to move a piece, its available moves will be highlighted in light blue.

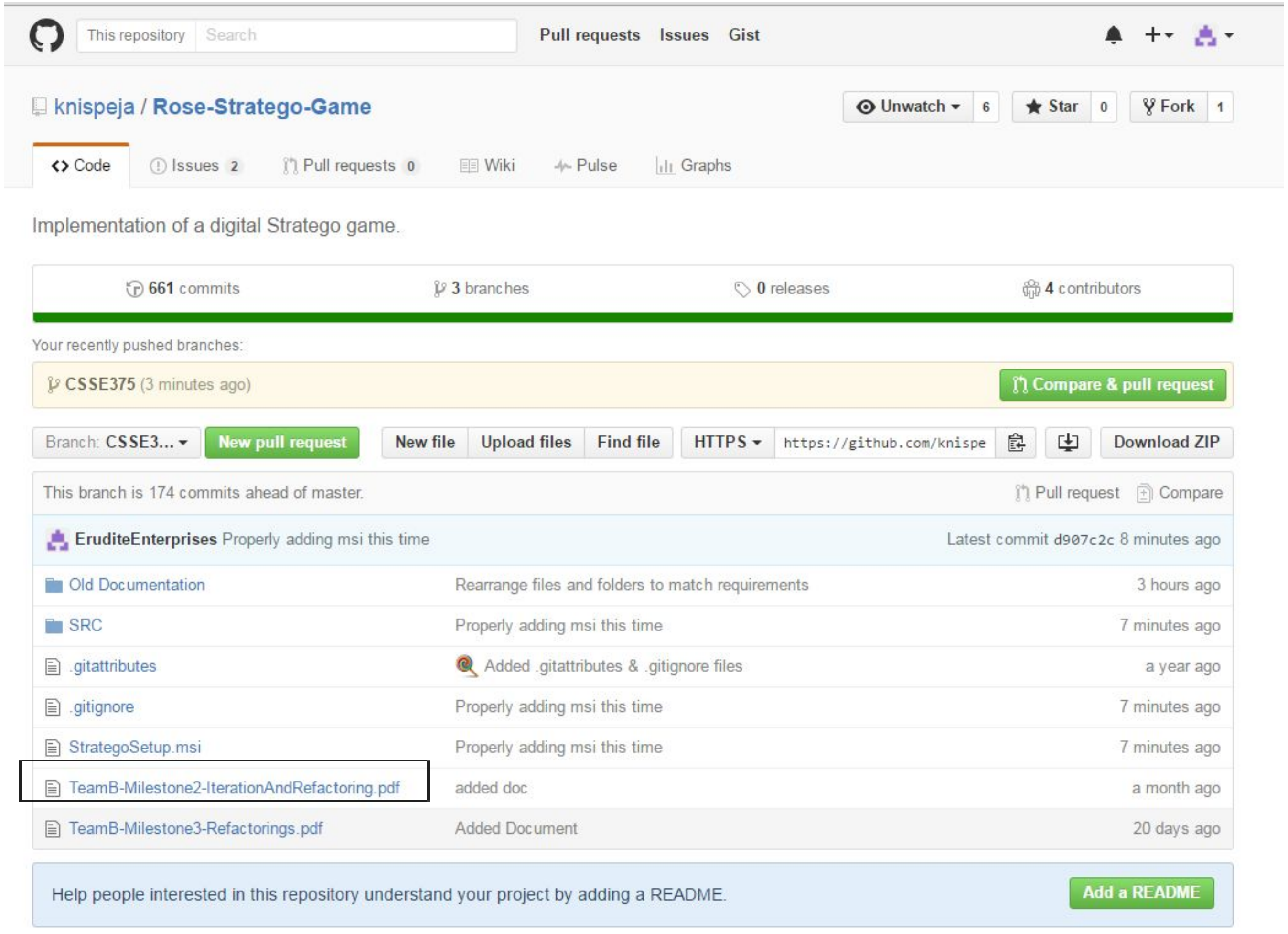


Installation, Configuration, and Maintenance Guide

Installation Guide:

User Installation:

Installing our Stratego game involves just a few easy steps. First, you need to download our installer from our GitHub at https://github.com/knispeja/Rose-Stratego-Game/tree/CSSE375.



Implementation of a digital Stratego game.

661 commits · 3 branches · 0 releases · 4 contributors

Your recently pushed branches:

CSSE375 (3 minutes ago)

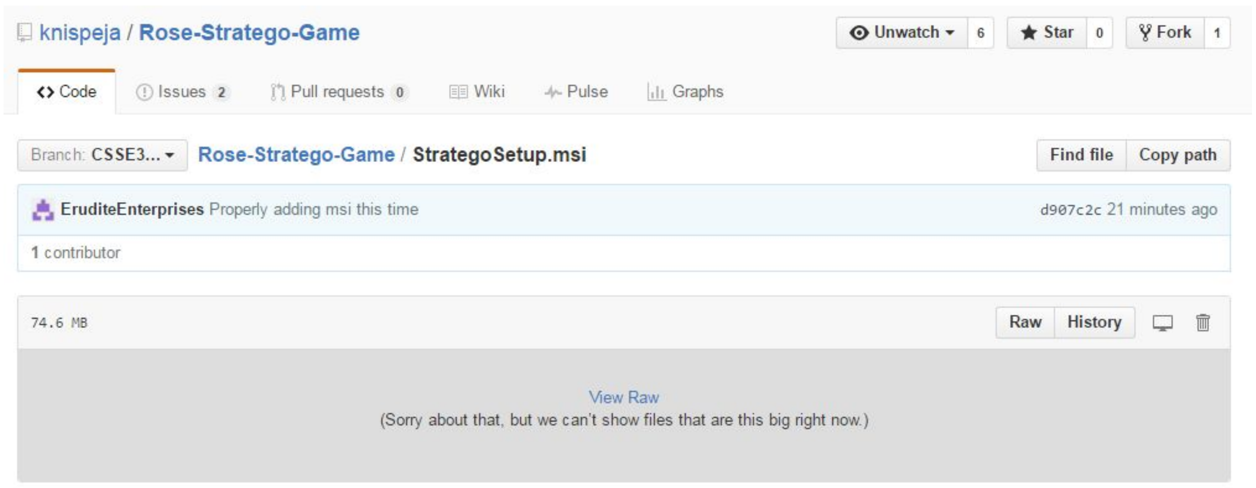
This branch is 174 commits ahead of master.

EruditeEnterprises Properly adding msi this time — Latest commit d907c2c 8 minutes ago

Old Documentation	Rearrange files and folders to match requirements	3 hours ago
SRC	Properly adding msi this time	7 minutes ago
.gitattributes	Added .gitattributes & .gitignore files	a year ago
.gitignore	Properly adding msi this time	7 minutes ago
StrategoSetup.msi	Properly adding msi this time	7 minutes ago
TeamB-Milestone2-IterationAndRefactoring.pdf	added doc	a month ago
TeamB-Milestone3-Refactorings.pdf	Added Document	20 days ago

Help people interested in this repository understand your project by adding a README.

You should see the page shown above. If not make sure that the branch in the top left is set to CSSE375. On this page, click the link to the StrategoSetup.msi link here.



Rose-Stratego-Game / StrategoSetup.msi

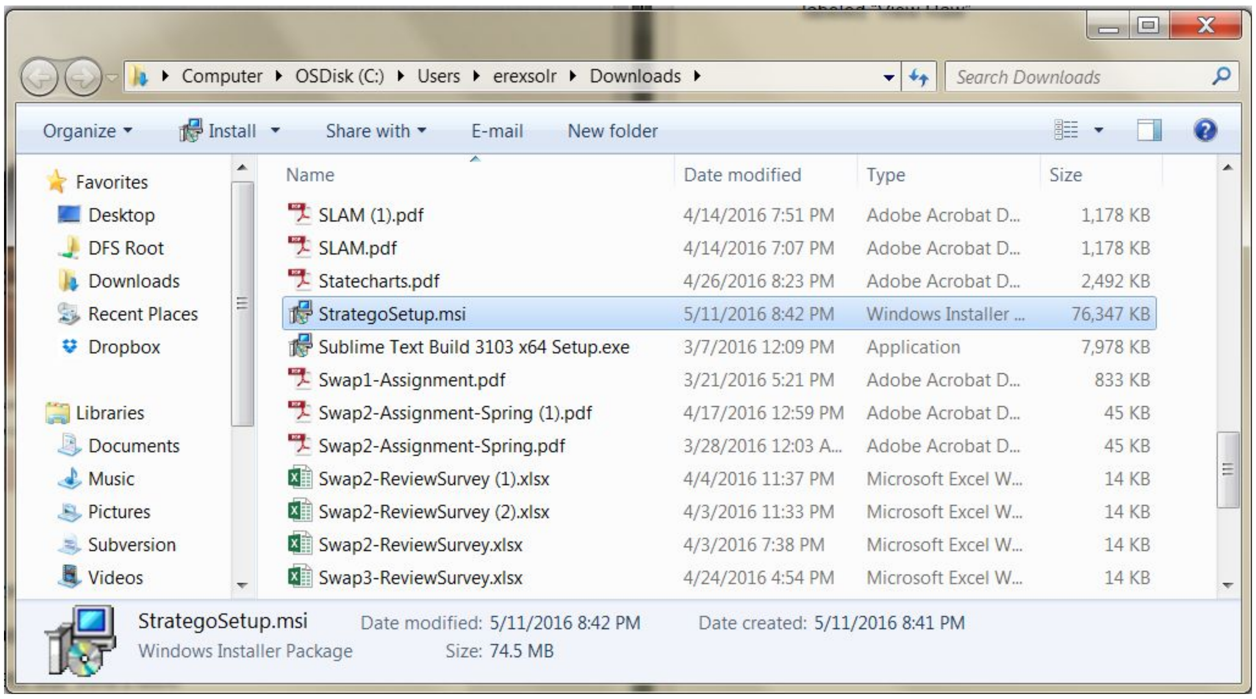
EruditeEnterprises Properly adding msi this time d907c2c 21 minutes ago

1 contributor

74.6 MB

View Raw
(Sorry about that, but we can't show files that are this big right now.)

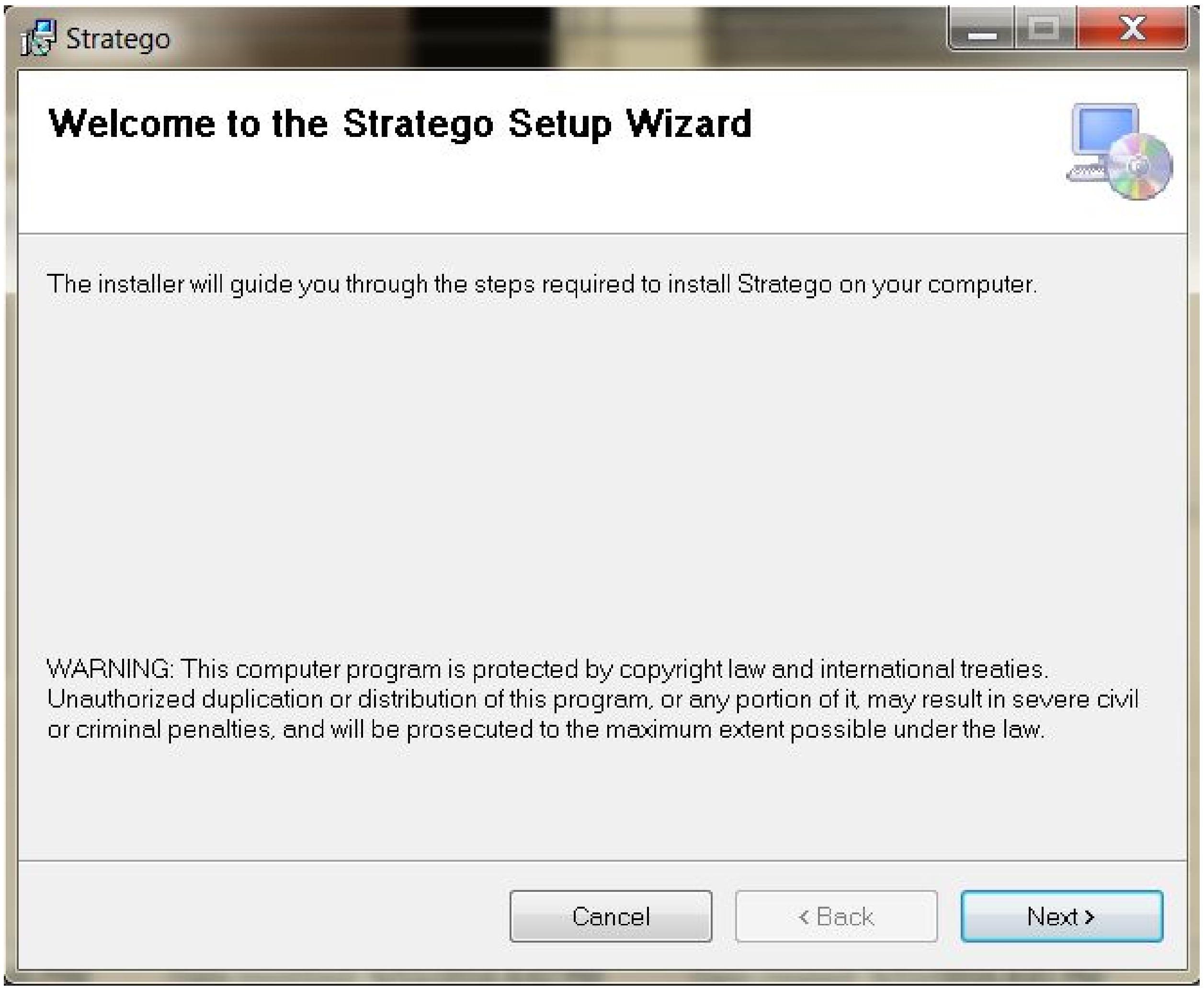
It should bring you to the above page where you can begin downloading by clicking on the link labeled “View Raw”.



Name	Date modified	Type	Size
SLAM (1).pdf	4/14/2016 7:51 PM	Adobe Acrobat D...	1,178 KB
SLAM.pdf	4/14/2016 7:07 PM	Adobe Acrobat D...	1,178 KB
Statecharts.pdf	4/26/2016 8:23 PM	Adobe Acrobat D...	2,492 KB
StrategoSetup.msi	5/11/2016 8:42 PM	Windows Installer ...	76,347 KB
Sublime Text Build 3103 x64 Setup.exe	3/7/2016 12:09 PM	Application	7,978 KB
Swap1-Assignment.pdf	3/21/2016 5:21 PM	Adobe Acrobat D...	833 KB
Swap2-Assignment-Spring (1).pdf	4/17/2016 12:59 PM	Adobe Acrobat D...	45 KB
Swap2-Assignment-Spring.pdf	3/28/2016 12:03 A...	Adobe Acrobat D...	45 KB
Swap2-ReviewSurvey (1).xlsx	4/4/2016 11:37 PM	Microsoft Excel W...	14 KB
Swap2-ReviewSurvey (2).xlsx	4/3/2016 11:33 PM	Microsoft Excel W...	14 KB
Swap2-ReviewSurvey.xlsx	4/3/2016 7:38 PM	Microsoft Excel W...	14 KB
Swap3-ReviewSurvey.xlsx	4/24/2016 4:54 PM	Microsoft Excel W...	14 KB

StrategoSetup.msi Date modified: 5/11/2016 8:42 PM Date created: 5/11/2016 8:41 PM
Windows Installer Package Size: 74.5 MB

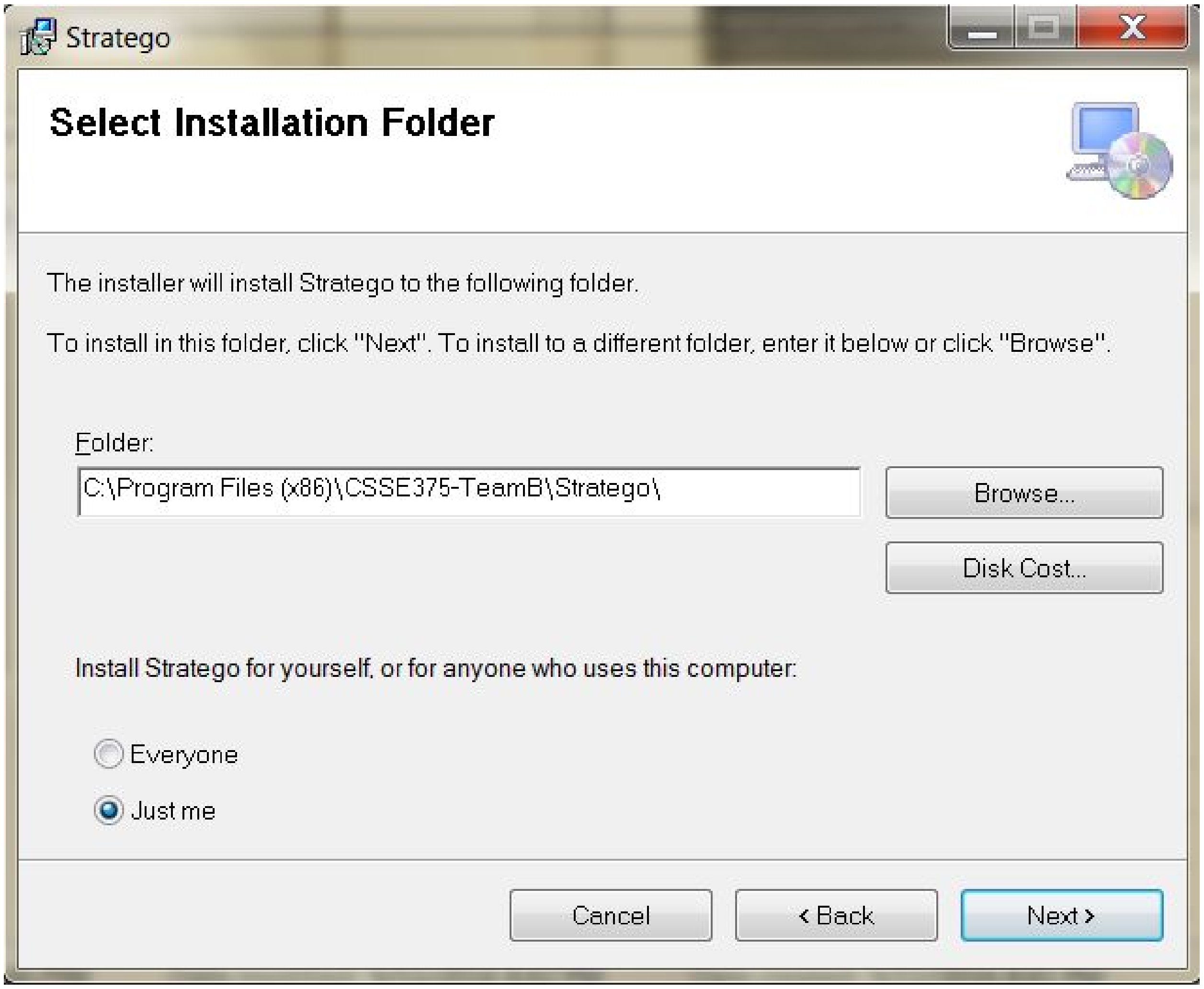
You can find the downloaded msi file in your default downloads folder. Double-click on the file to run the installer.


Welcome to the Stratego Setup Wizard

The installer will guide you through the steps required to install Stratego on your computer.

WARNING: This computer program is protected by copyright law and international treaties. Unauthorized duplication or distribution of this program, or any portion of it, may result in severe civil or criminal penalties, and will be prosecuted to the maximum extent possible under the law.

This is the window that should show up when the installer is run. You may instead get a window that asks you if you want to run the installer. If it shows up, click the “Run” button. Once you see the above window, click the “Next” button.


Stratego

Select Installation Folder

The installer will install Stratego to the following folder.

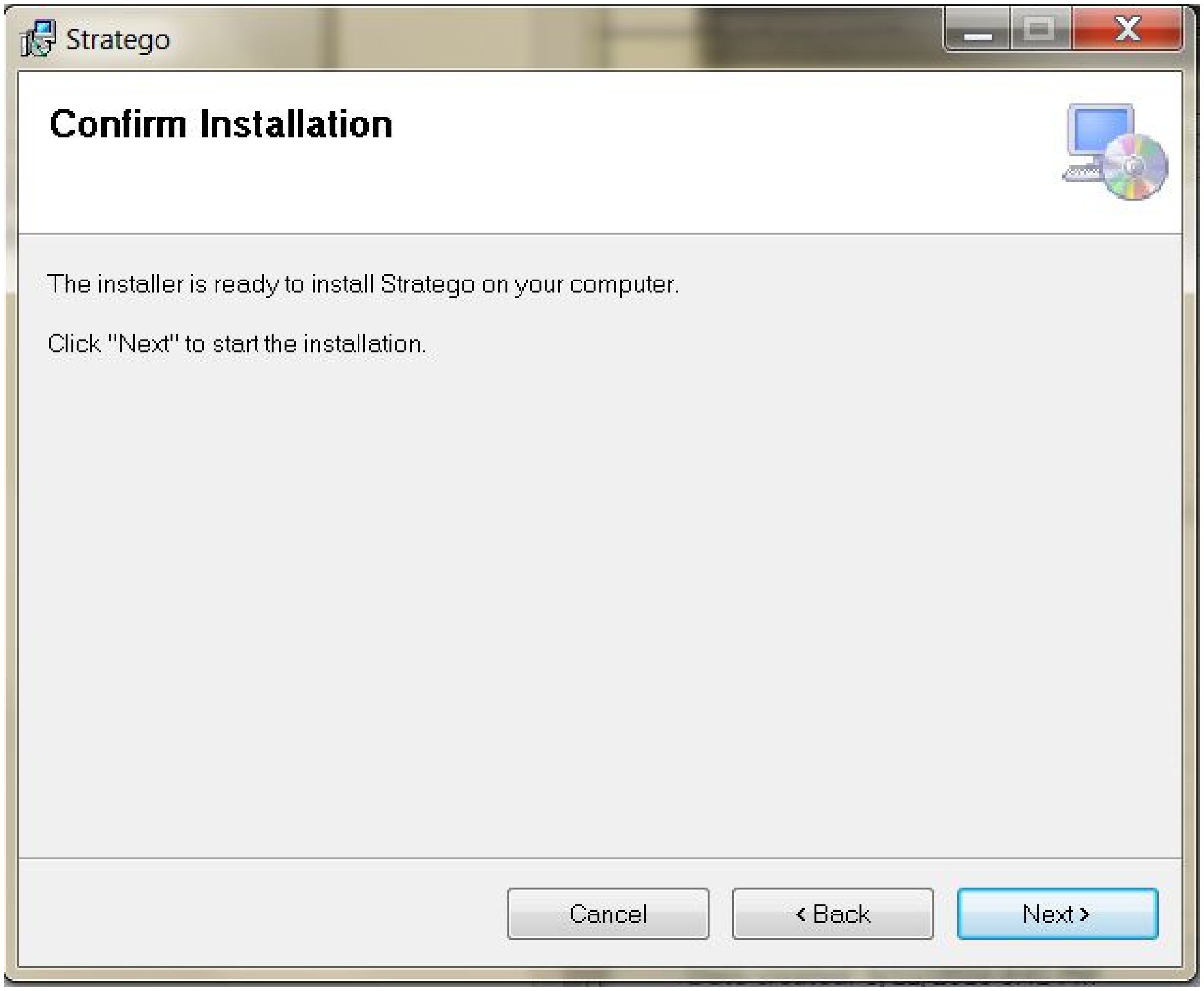
To install in this folder, click "Next". To install to a different folder, enter it below or click "Browse".

Folder:
C:\Program Files (x86)\CSSE375-TeamB\Stratego\

Install Stratego for yourself, or for anyone who uses this computer:

Everyone [✓ Just me]

On the next screen you have the option to choose a different installation location. By default the game will install to a folder in your program files location. Additionally, if you have multiple users on your computer, you can choose to give access to the game to all users by selecting the bubble labeled “Everyone”. Keeping the “Just me” option selected will only give access to the current user. Once you have made any changes necessary, continue with the “Next” button.

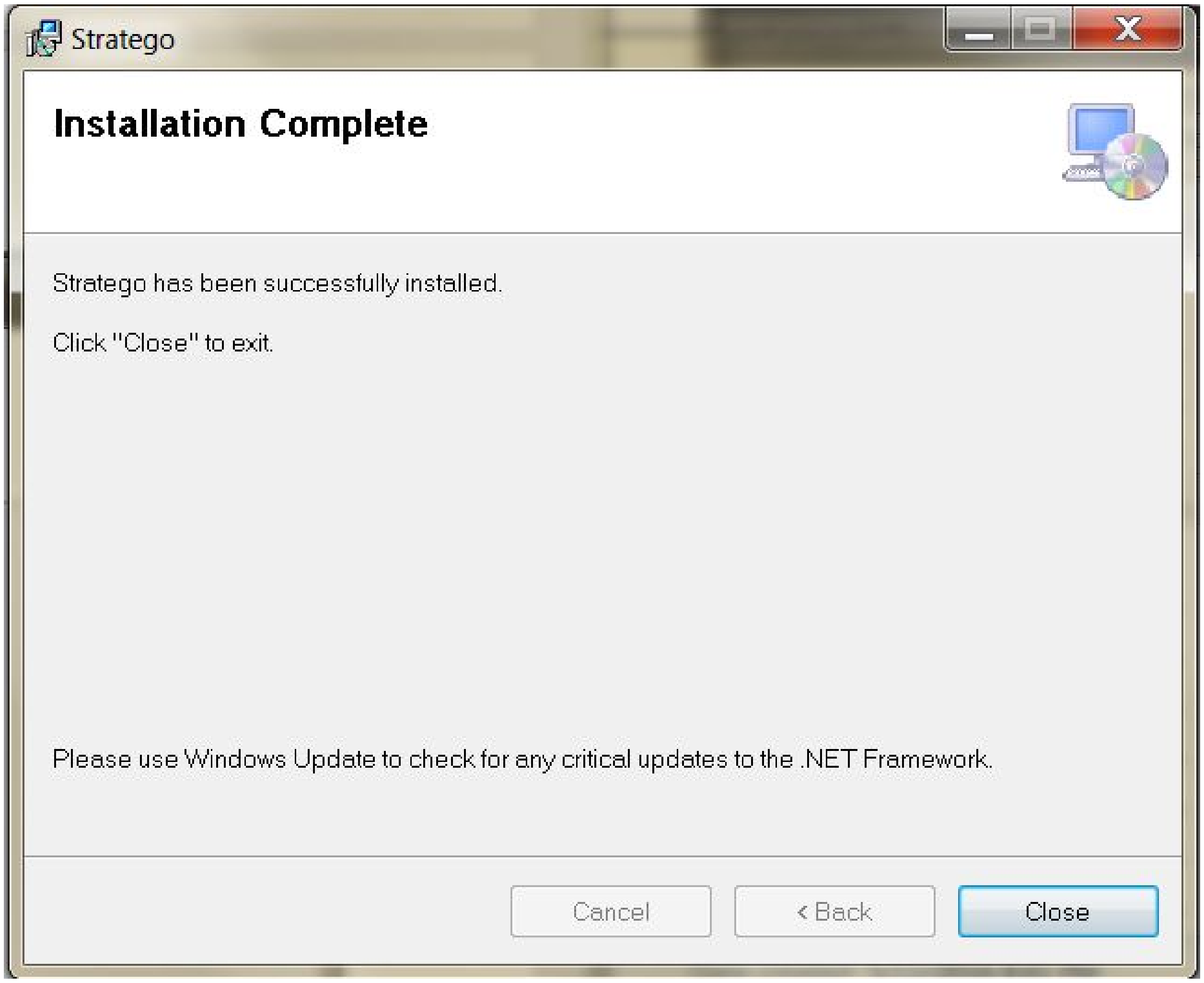

Stratego

Confirm Installation

The installer is ready to install Stratego on your computer.

Click "Next" to start the installation.

Click the “Next” button on the next page to start the installation. It should take at most a few minutes.



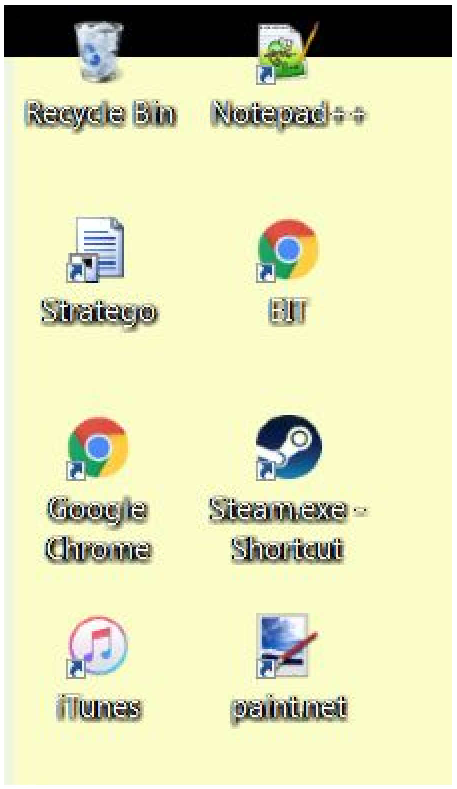
Installation Complete

Stratego has been successfully installed.

Click "Close" to exit.

Please use Windows Update to check for any critical updates to the .NET Framework.

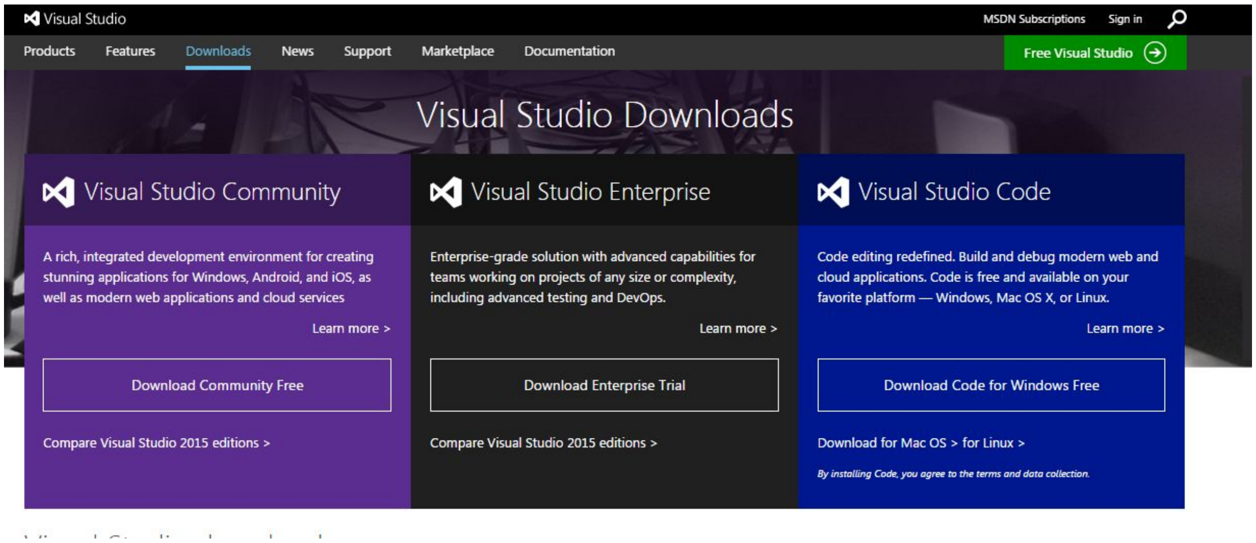
After the installation is complete, you can just close the window.



The installation should have placed a shortcut on your desktop and in your start menu. You can click either to start the game now.

Developer/Maintainer Installation Guide

Before installing our code, you need to install a few other things.



Visual Studio Downloads

Visual Studio Community

A rich, integrated development environment for creating stunning applications for Windows, Android, and iOS, as well as modern web applications and cloud services

Learn more >

Download Community Free

Compare Visual Studio 2015 editions >

Visual Studio Enterprise

Enterprise-grade solution with advanced capabilities for teams working on projects of any size or complexity, including advanced testing and DevOps.

Learn more >

Download Enterprise Trial

Compare Visual Studio 2015 editions >

Visual Studio Code

Code editing redefined. Build and debug modern web and cloud applications. Code is free and available on your favorite platform — Windows, Mac OS X, or Linux.

Learn more >

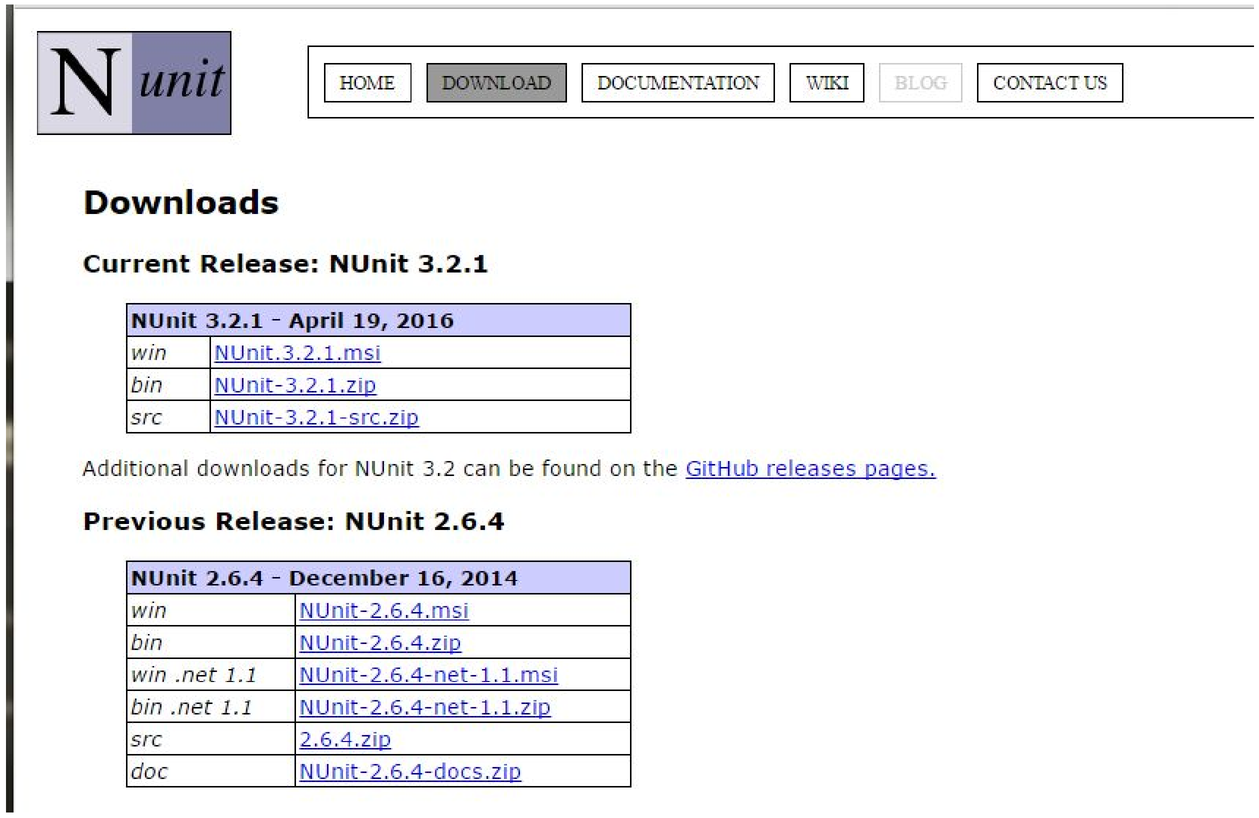
Download Code for Windows Free

Download for Mac OS > for Linux >

By installing Code, you agree to the terms and data collection.

First, you need to have either Visual Studio Community 2015 or Visual Studio Enterprise 2015. You can download that at https://www.visualstudio.com/en-us/downloads/download-visual-studio-vs.aspx.

Once you have Visual Studio, you need to have the Nunit plugin for Visual Studio to be able to run our tests.



Downloads

Current Release: NUnit 3.2.1

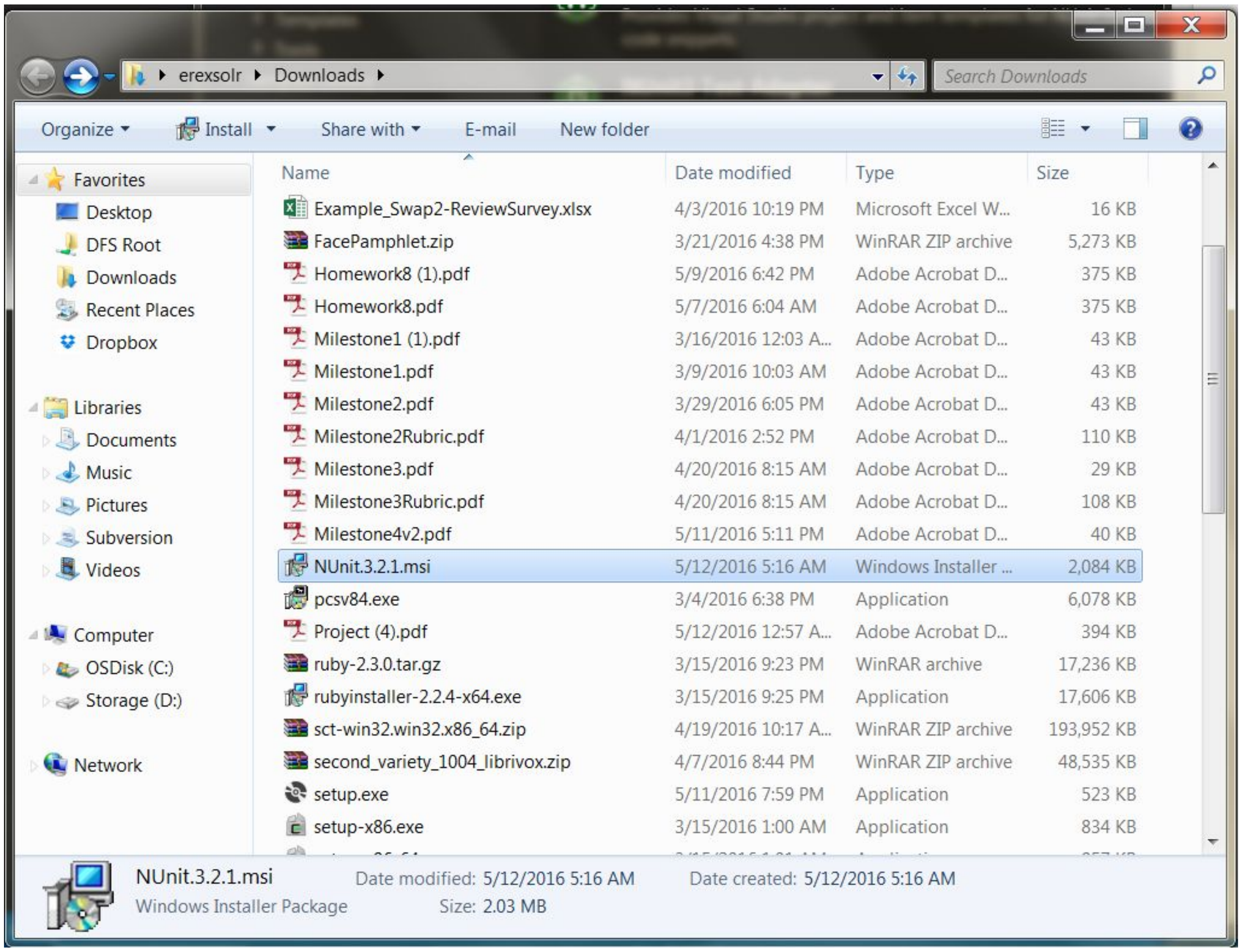
NUnit 3.2.1 - April 19, 2016	
win	NUnit.3.2.1.msi
bin	NUnit-3.2.1.zip
src	NUnit-3.2.1-src.zip

Additional downloads for NUnit 3.2 can be found on the GitHub releases pages.

Previous Release: NUnit 2.6.4

NUnit 2.6.4 - December 16, 2014	
win	NUnit-2.6.4.msi
bin	NUnit-2.6.4.zip
win .net 1.1	NUnit-2.6.4-net-1.1.msi
bin .net 1.1	NUnit-2.6.4-net-1.1.zip
src	2.6.4.zip
doc	NUnit-2.6.4-docs.zip

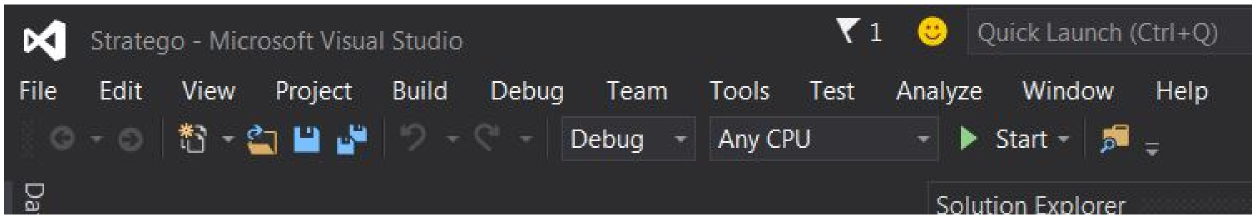
On the NUnit download page (http://www.nunit.org/index.php?p=download), select the msi download.



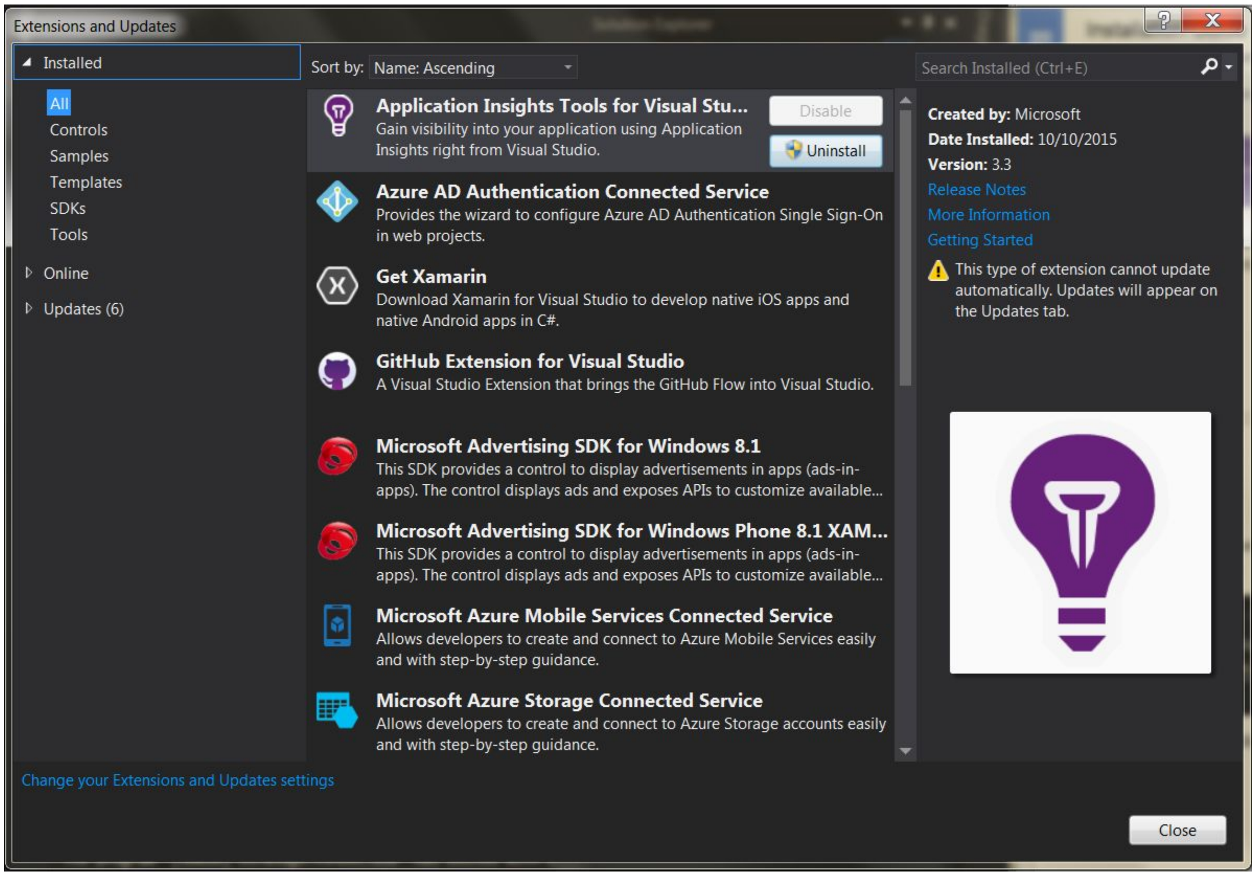
Name	Date modified	Type	Size
Example_Swap2-ReviewSurvey.xlsx	4/3/2016 10:19 PM	Microsoft Excel W...	16 KB
FacePamphlet.zip	3/21/2016 4:38 PM	WinRAR ZIP archive	5,273 KB
Homework8 (1).pdf	5/9/2016 6:42 PM	Adobe Acrobat D...	375 KB
Homework8.pdf	5/7/2016 6:04 AM	Adobe Acrobat D...	375 KB
Milestone1 (1).pdf	3/16/2016 12:03 A...	Adobe Acrobat D...	43 KB
Milestone1.pdf	3/9/2016 10:03 AM	Adobe Acrobat D...	43 KB
Milestone2.pdf	3/29/2016 6:05 PM	Adobe Acrobat D...	43 KB
Milestone2Rubric.pdf	4/1/2016 2:52 PM	Adobe Acrobat D...	110 KB
Milestone3.pdf	4/20/2016 8:15 AM	Adobe Acrobat D...	29 KB
Milestone3Rubric.pdf	4/20/2016 8:15 AM	Adobe Acrobat D...	108 KB
Milestone4v2.pdf	5/11/2016 5:11 PM	Adobe Acrobat D...	40 KB
NUnit.3.2.1.msi	5/12/2016 5:16 AM	Windows Installer ...	2,084 KB
pcsv84.exe	3/4/2016 6:38 PM	Application	6,078 KB
Project (4).pdf	5/12/2016 12:57 A...	Adobe Acrobat D...	394 KB
ruby-2.3.0.tar.gz	3/15/2016 9:23 PM	WinRAR archive	17,236 KB
rubyinstaller-2.2.4-x64.exe	3/15/2016 9:25 PM	Application	17,606 KB
sct-win32.win32.x86_64.zip	4/19/2016 10:17 A...	WinRAR ZIP archive	193,952 KB
second_variety_1004_librivox.zip	4/7/2016 8:44 PM	WinRAR ZIP archive	48,535 KB
setup.exe	5/11/2016 7:59 PM	Application	523 KB
setup-x86.exe	3/15/2016 1:00 AM	Application	834 KB

NUnit.3.2.1.msi Date modified: 5/12/2016 5:16 AM Date created: 5/12/2016 5:16 AM
Windows Installer Package Size: 2.03 MB

Once the file is downloaded, run it from your downloads file. Follow the prompts to download NUnit.



Next we need an adapter for NUnit in Visual Studio. In Visual Studio, open the tools menu and select Extensions and Updates.



Application Insights Tools for Visual Stu...
Gain visibility into your application using Application Insights right from Visual Studio.

Azure AD Authentication Connected Service
Provides the wizard to configure Azure AD Authentication Single Sign-On in web projects.

Get Xamarin
Download Xamarin for Visual Studio to develop native iOS apps and native Android apps in C#.

GitHub Extension for Visual Studio
A Visual Studio Extension that brings the GitHub Flow into Visual Studio.

Microsoft Advertising SDK for Windows 8.1
This SDK provides a control to display advertisements in apps (ads-in-apps). The control displays ads and exposes APIs to customize available...

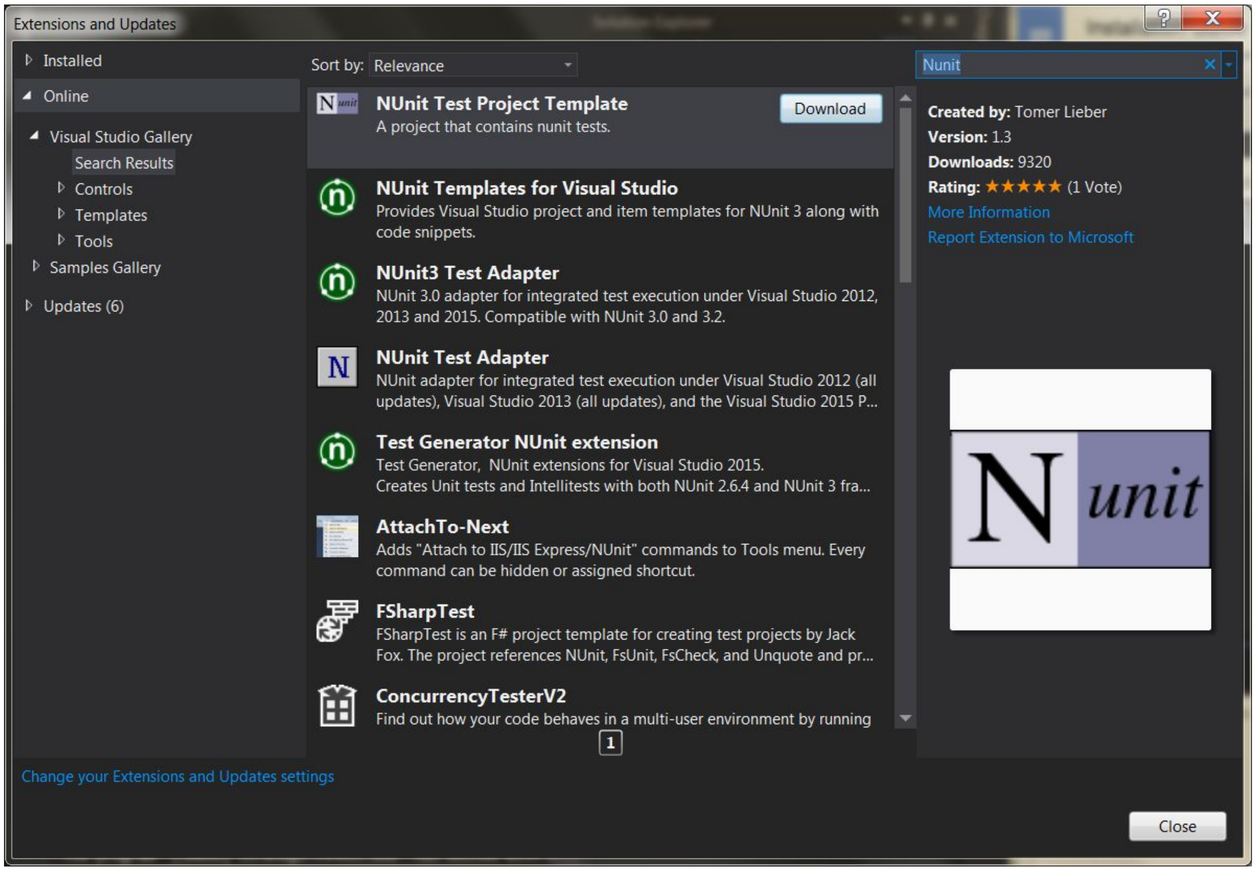
Microsoft Advertising SDK for Windows Phone 8.1 XAM...
This SDK provides a control to display advertisements in apps (ads-in-apps). The control displays ads and exposes APIs to customize available...

Microsoft Azure Mobile Services Connected Service
Allows developers to create and connect to Azure Mobile Services easily and with step-by-step guidance.

Microsoft Azure Storage Connected Service
Allows developers to create and connect to Azure Storage accounts easily and with step-by-step guidance.

Created by: Microsoft
Date Installed: 10/10/2015
Version: 3.3
Release Notes
More Information
Getting Started
⚠ This type of extension cannot update automatically. Updates will appear on the Updates tab.

This will open a new menu where you can update or install extensions for visual studio.



NUnit Test Project Template
A project that contains nunit tests.

NUnit Templates for Visual Studio
Provides Visual Studio project and item templates for NUnit 3 along with code snippets.

NUnit3 Test Adapter
NUnit 3.0 adapter for integrated test execution under Visual Studio 2012, 2013 and 2015. Compatible with NUnit 3.0 and 3.2.

NUnit Test Adapter
NUnit adapter for integrated test execution under Visual Studio 2012 (all updates), Visual Studio 2013 (all updates), and the Visual Studio 2015 P...

Test Generator NUnit extension
Test Generator, NUnit extensions for Visual Studio 2015.
Creates Unit tests and Intellitests with both NUnit 2.6.4 and NUnit 3 fra...

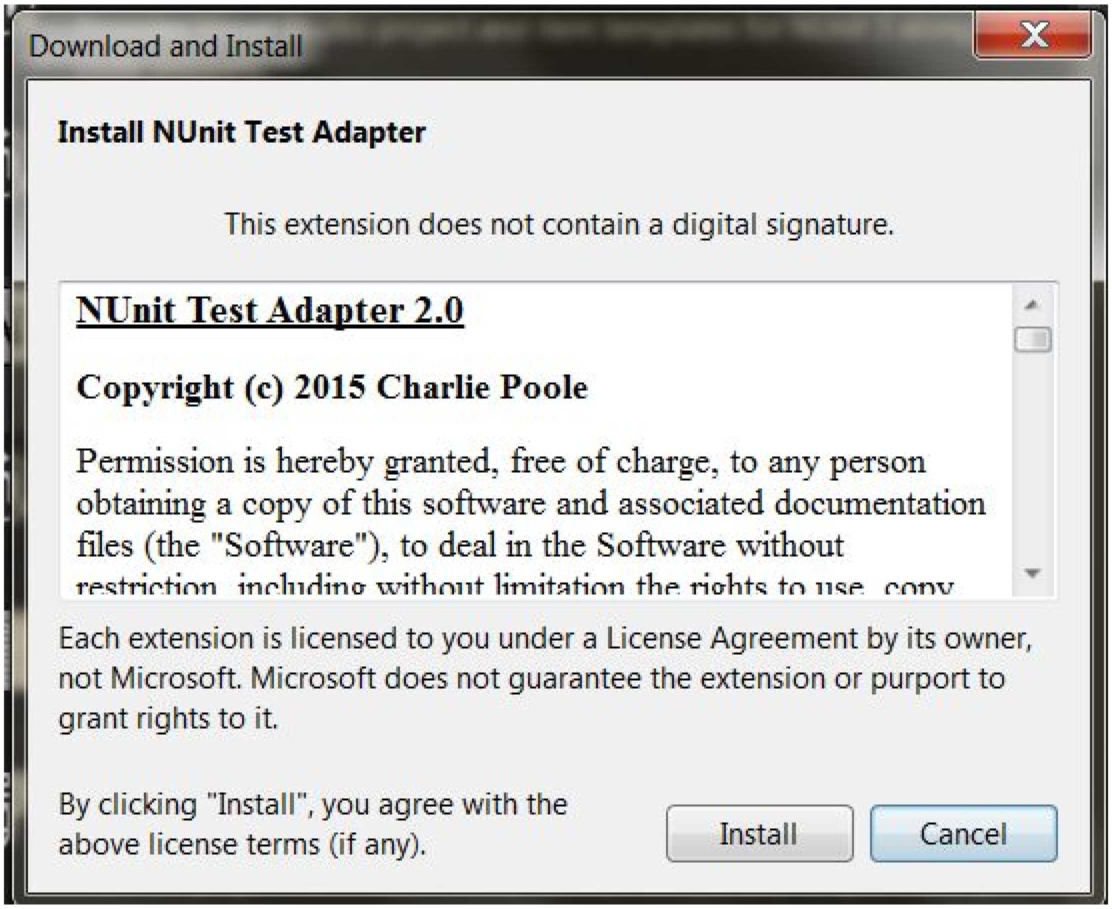
AttachTo-Next
Adds "Attach to IIS/IIS Express/NUnit" commands to Tools menu. Every command can be hidden or assigned shortcut.

FSharpTest
FSharpTest is an F# project template for creating test projects by Jack Fox. The project references NUnit, FsUnit, FsCheck, and Unquote and pr...

ConcurrencyTesterV2
Find out how your code behaves in a multi-user environment by running

Created by: Tomer Lieber
Version: 1.3
Downloads: 9320
Rating: ★★★★★ (1 Vote)
More Information
Report Extension to Microsoft

After selection the Online section on the left, type Nunit into the search box on the right. After the search is complete you should see NUnit Test Adapter as the fourth item. Select it and click the “Download” button.



Download and Install

Install NUnit Test Adapter

This extension does not contain a digital signature.

NUnit Test Adapter 2.0

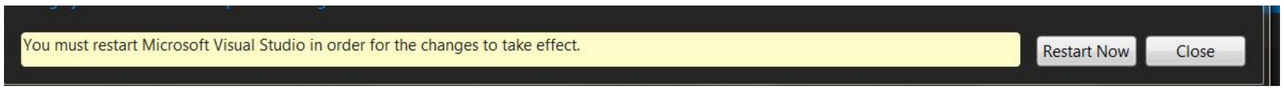
Copyright (c) 2015 Charlie Poole

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy

Each extension is licensed to you under a License Agreement by its owner, not Microsoft. Microsoft does not guarantee the extension or purport to grant rights to it.

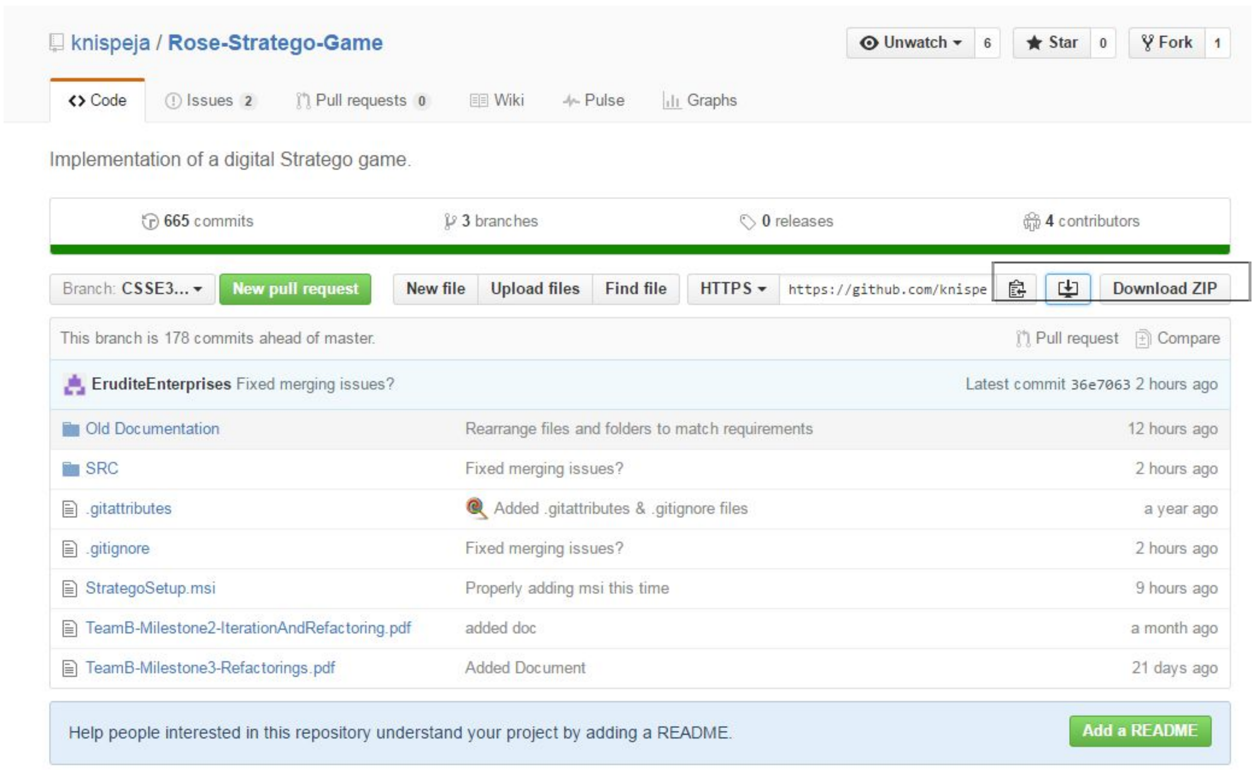
By clicking "Install", you agree with the above license terms (if any).

Click the “Install” button. After the installation is finished you will be prompted to restart Visual Studio. Click the “Restart Now” button to do so. After Visual Studio restarts, NUnit should be functional.



You must restart Microsoft Visual Studio in order for the changes to take effect.

Now that NUnit is installed, all that’s left is to get the source code.



Implementation of a digital Stratego game.

665 commits | 3 branches | 0 releases | 4 contributors

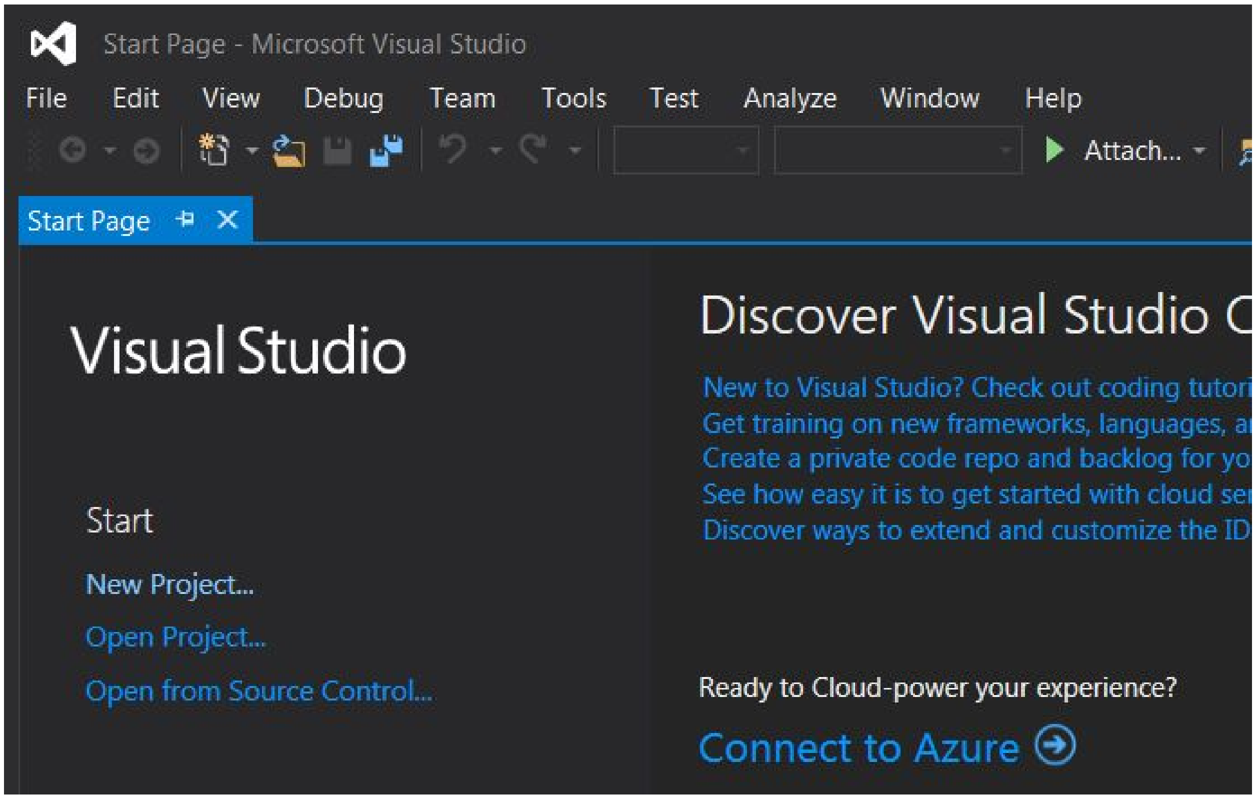
This branch is 178 commits ahead of master.

EruditeEnterprises Fixed merging issues? Latest commit 36e7063 2 hours ago

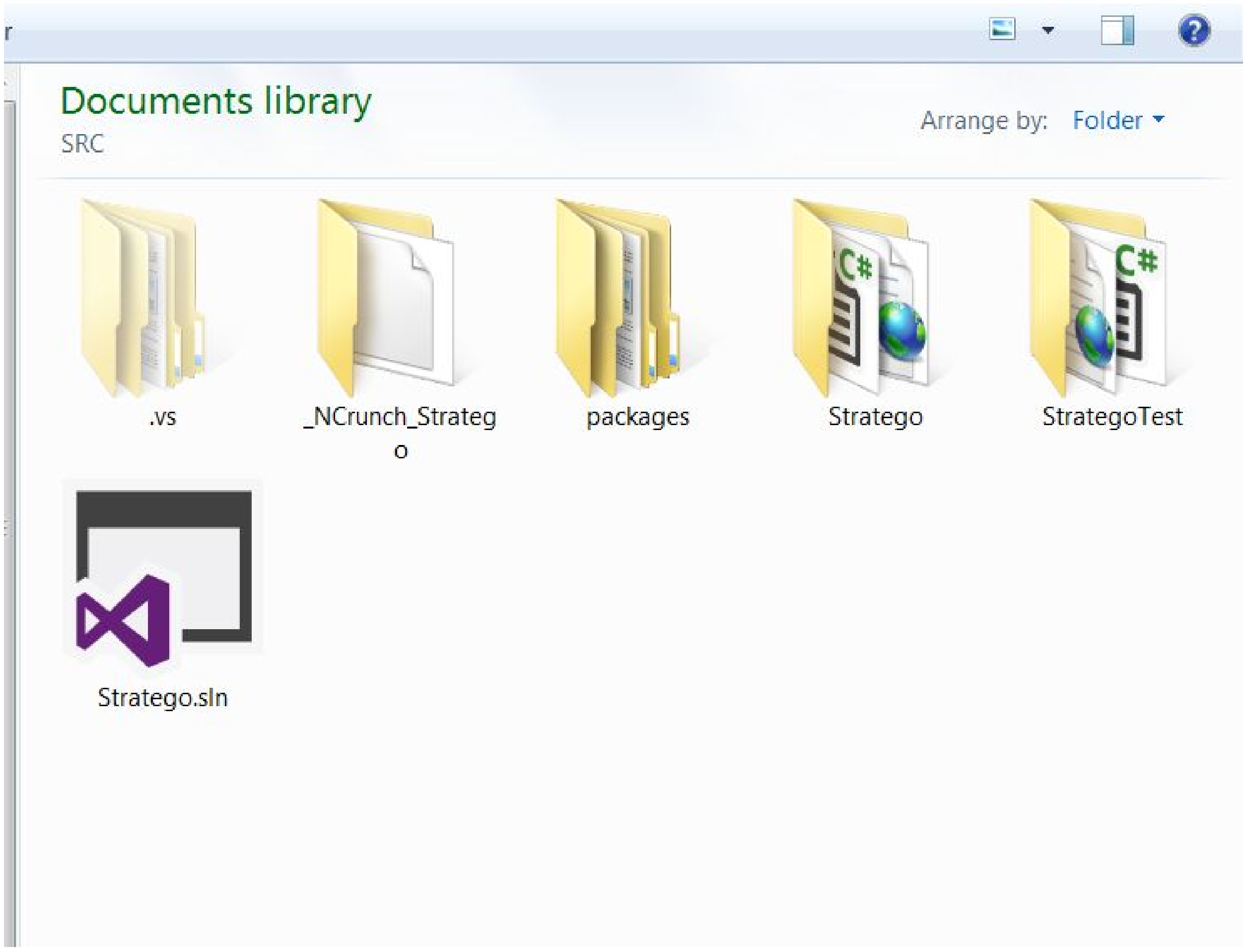
Old Documentation	Rearrange files and folders to match requirements	12 hours ago
SRC	Fixed merging issues?	2 hours ago
.gitattributes	Added .gitattributes & .gitignore files	a year ago
.gitignore	Fixed merging issues?	2 hours ago
StrategoSetup.msi	Properly adding msi this time	9 hours ago
TeamB-Milestone2-IterationAndRefactoring.pdf	added doc	a month ago
TeamB-Milestone3-Refactorings.pdf	Added Document	21 days ago

Help people interested in this repository understand your project by adding a README.

At https://github.com/knispeja/Rose-Stratego-Game/tree/CSSE375, you can use the URL to in a Git console, clone the repository, or download the zip to obtain the source code. The three highlighted buttons above are used for each of the methods respectively.


Start Page - Microsoft Visual Studio
File
Edit
View
Debug
Team
Tools
Test
Analyze
Window
Help
Attach...
Start Page
Visual Studio
Start
New Project...
Open Project...
Open from Source Control...
Discover Visual Studio C
New to Visual Studio? Check out coding tutori
Get training on new frameworks, languages, a
Create a private code repo and backlog for yo
See how easy it is to get started with cloud se
Discover ways to extend and customize the ID
Ready to Cloud-power your experience?
Connect to Azure

Once you have the source code, open visual studio and select Open Project on the start page.



Navigate to the SRC folder of the downloaded repository and select Stratego.sln to open the project. From here you have everything you need to develop or maintain for our Stratego game.

Maintenance Guide:

Help-Desk & Tech Support:

Refer to the Troubleshooting section below. Note that issues related to the Planned Features may not have solutions. Make notes of frequent issues and issues that do not have solutions for use by the maintainers.

Troubleshooting:

Problem	Solution
Application does not start or seems to behave improperly after starting	Ensure that application is being run on a compatible operating system (Windows 7/8/8.1/10). Attempt to start application again. If problem persists, reinstall application.
Application crashes or freezes (does not follow a specific user action or crash/freeze cannot be duplicated)	Restart the application.
Application crashes or freezes upon loading a saved game or layout	Start new game, save game or layout, restart application, and attempt to load newly created save. If new save loads, save causing the crash or freeze is corrupt. Otherwise, no solution; avoid using saves and make note for maintainers.
Application crashes or freezes at the beginning of AI turn, during AI turn, or at the end of AI turn	Avoid playing against AI and contact maintainers.
Application crashes or freezes after other action (not listed above)	Avoid action and contact maintainers.
AI makes repeating moves	AI is not fully implemented, ignore.

Maintainer:

Tools:

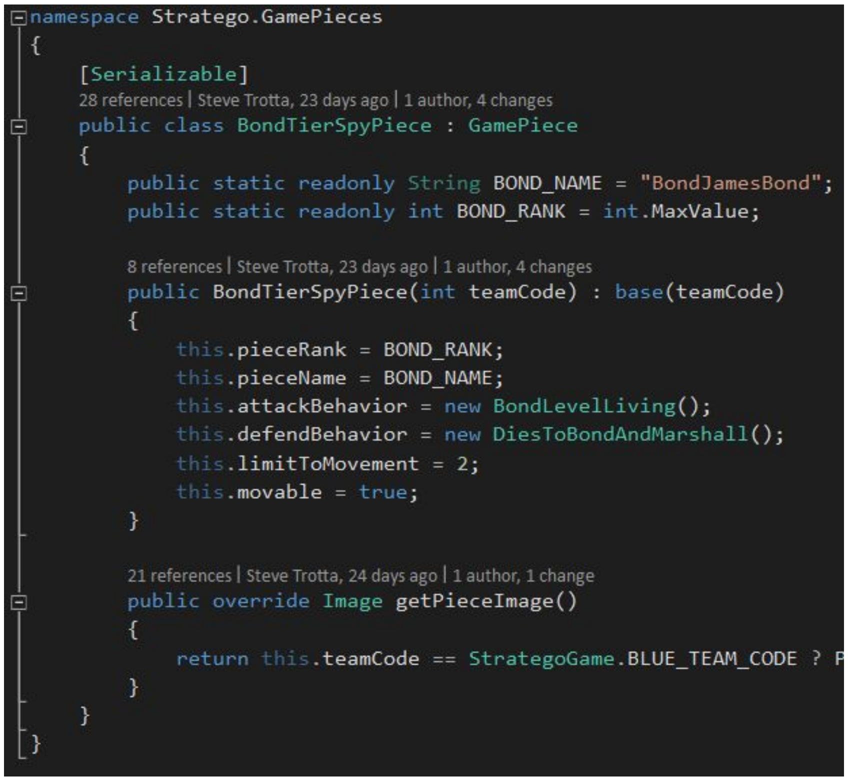
Maintenance of this project requires the software outlined in the Developer/Maintainer Installation Guide. Updates to the source code should be pushed to the project Gitlab -- note that new builds must pass regression tests to be pushed.

Adding Pieces:

To create a new piece, create a new class in the GamePieces folder that extends GamePiece. Then, set public static readonly fields for the piece's name and rank (used in battle behaviors). In the constructor, set the fields as follows:

pieceRank: previously defined static rank field
pieceName: previously defined static name field
attackBehavior: a battle behavior (used when the piece moves onto another piece)
defendBehavior: a battle behavior (used when another piece moves onto the piece)
limitToMovement: an int (specifies the maximum number of spaces the piece may move)
Movable: a boolean (specifies whether the piece can move by default)

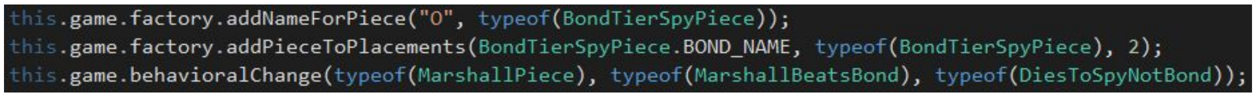
Next, override getPieceImage and return the filepath to the image you wish to use for the piece in-game



```
namespace Stratego.GamePieces
{
    [Serializable]
    28 references | Steve Trotta, 23 days ago | 1 author, 4 changes
    public class BondTierSpyPiece : GamePiece
    {
        public static readonly String BOND_NAME = "BondJamesBond";
        public static readonly int BOND_RANK = int.MaxValue;

        8 references | Steve Trotta, 23 days ago | 1 author, 4 changes
        public BondTierSpyPiece(int teamCode) : base(teamCode)
        {
            this.pieceRank = BOND_RANK;
            this.pieceName = BOND_NAME;
            this.attackBehavior = new BondLevelLiving();
            this.defendBehavior = new DiesToBondAndMarshall();
            this.limitToMovement = 2;
            this.movable = true;
        }

        21 references | Steve Trotta, 24 days ago | 1 author, 1 change
        public override Image getPieceImage()
        {
            return this.teamCode == StrategoGame.BLUE_TEAM_CODE ? P
        }
    }
}
```

.

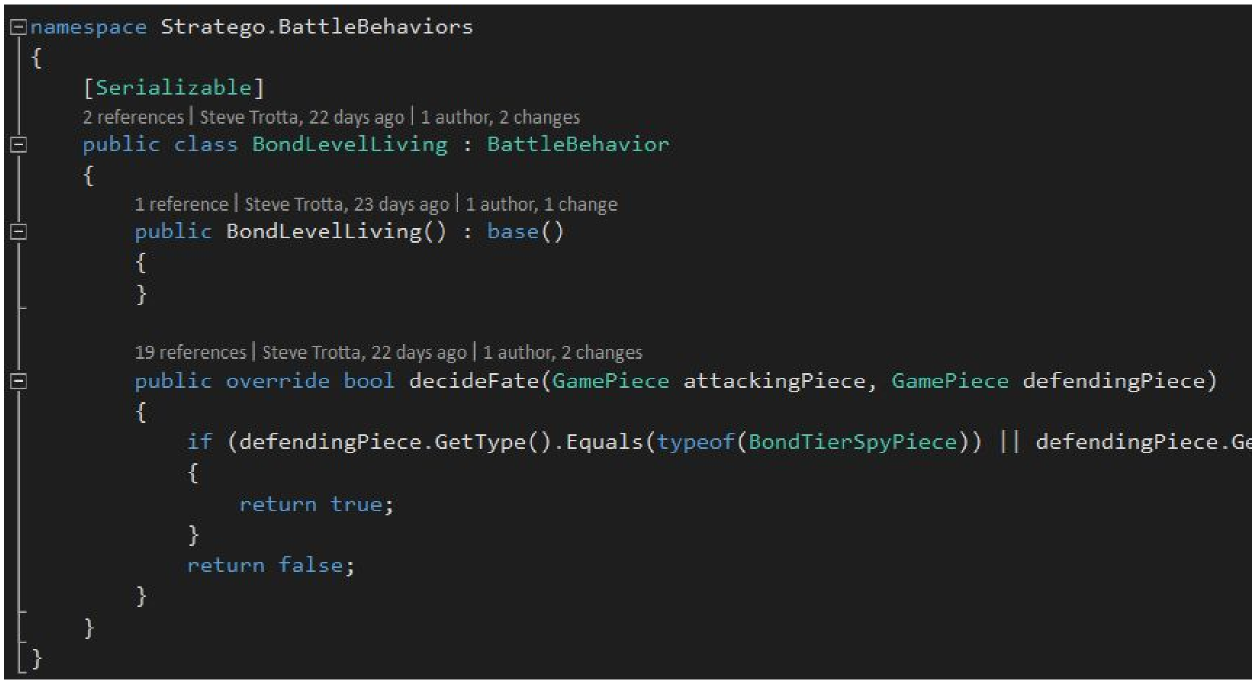
Finally, to be able to use the piece in-game, calls must be made to to the GamePieceFactory's addNameForPiece and addPieceToPlacements methods. Calls to behavioralChange can also be used in order to update the behaviors of other pieces in relation to the new piece.



```
this.game.factory.addNameForPiece("O", typeof(BondTierSpyPiece));
this.game.factory.addPieceToPlacements(BondTierSpyPiece.BOND_NAME, typeof(BondTierSpyPiece), 2);
this.game.behavioralChange(typeof(MarshallPiece), typeof(MarshallBeatsBond), typeof(DiesToSpyNotBond));
```

Adding Piece Battle Behaviors:

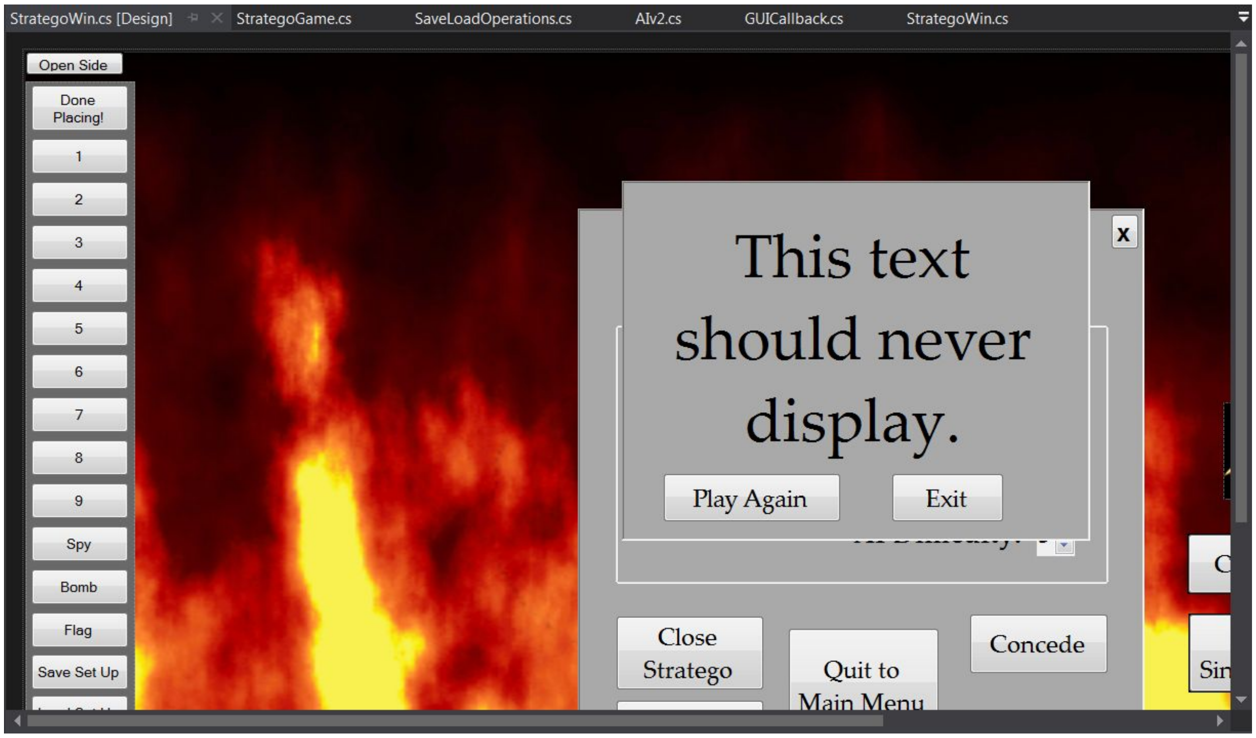
To add a battle behavior, simply create a new class in the BattleBehaviors folder that extends BattleBehavior and override the decideFate method. Any time this method is called, the defending piece (the first parameter) will be removed from the board if the method returns true; otherwise the attacking piece (the second parameter) will be removed.



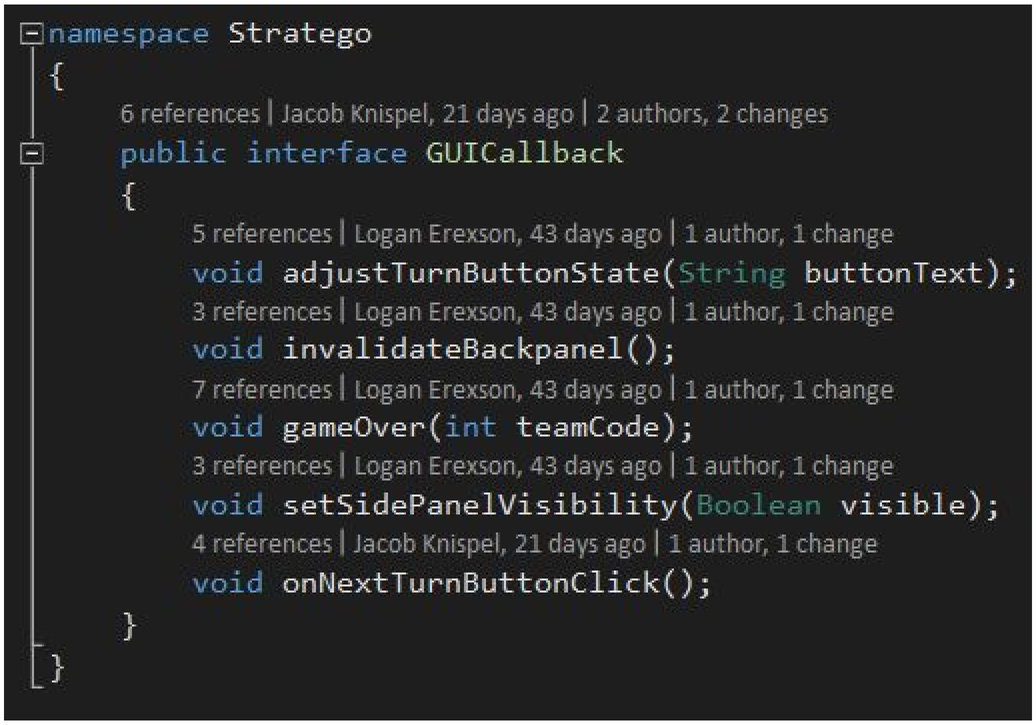
```
namespace Stratego.BattleBehaviors
{
    [Serializable]
    2 references | Steve Trotta, 22 days ago | 1 author, 2 changes
    public class BondLevelLiving : BattleBehavior
    {
        1 reference | Steve Trotta, 23 days ago | 1 author, 1 change
        public BondLevelLiving() : base()
        {
        }

        19 references | Steve Trotta, 22 days ago | 1 author, 2 changes
        public override bool decideFate(GamePiece attackingPiece, GamePiece defendingPiece)
        {
            if (defendingPiece.GetType().Equals(typeof(BondTierSpyPiece)) || defendingPiece.Ge
            {
                return true;
            }
            return false;
        }
    }
}
```

Making Changes to the GUI:

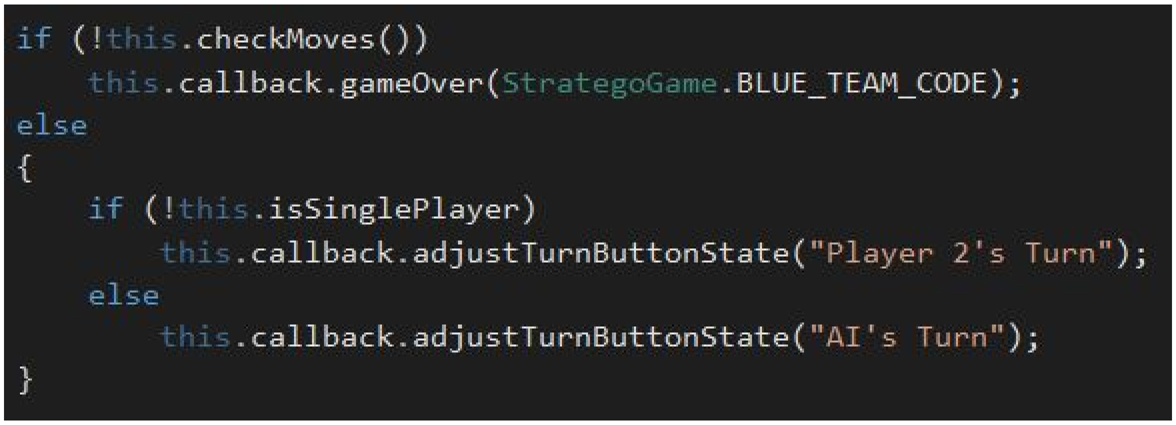
Any changes to the GUI should be confined to the StrategoWin class when possible. Because the project is made in Visual Studio, it is also possible to use the StrategoWin designer file to drag and drop components.



If a GUI element needs to be called from within the code, add the desired method to the GUICallback interface and implement it in StrategoWin; this will allow the method to be called from StrategoGame (which handles the main game algorithms).



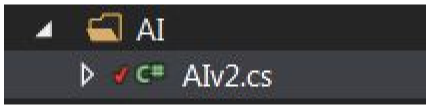
```
namespace Stratego
{
    6 references | Jacob Knispel, 21 days ago | 2 authors, 2 changes
    public interface GUICallback
    {
        5 references | Logan Erexson, 43 days ago | 1 author, 1 change
        void adjustTurnButtonState(String buttonText);
        3 references | Logan Erexson, 43 days ago | 1 author, 1 change
        void invalidateBackpanel();
        7 references | Logan Erexson, 43 days ago | 1 author, 1 change
        void gameOver(int teamCode);
        3 references | Logan Erexson, 43 days ago | 1 author, 1 change
        void setSidePanelVisibility(Boolean visible);
        4 references | Jacob Knispel, 21 days ago | 1 author, 1 change
        void onNextTurnButtonClick();
    }
}
```



```
if (!this.checkMoves())
    this.callback.gameOver(StrategoGame.BLUE_TEAM_CODE);
else
{
    if (!this.isSinglePlayer)
        this.callback.adjustTurnButtonState("Player 2's Turn");
    else
        this.callback.adjustTurnButtonState("AI's Turn");
}
```

Making Changes to the AI:

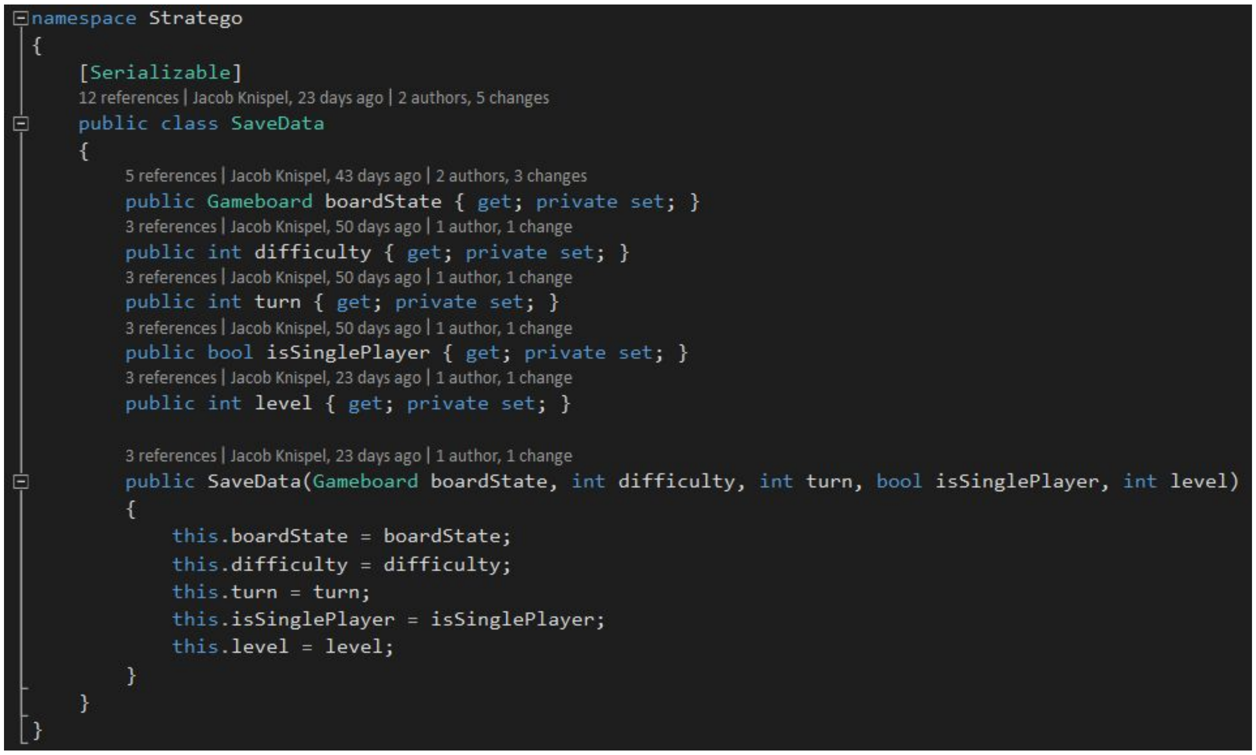
Changes to the AI should be made within the AI folder.



Any straightforward changes to the move-finding algorithm should be implemented in or called from the findMove method.

Adding Save/Load Functionality to an Object:

To add saving functionality to an object, the object must be added to the SaveData class. In addition to setting the SaveData's fields to track the new object, the object must be serializeable.



```
namespace Stratego
{
    [Serializable]
    12 references | Jacob Knispel, 23 days ago | 2 authors, 5 changes
    public class SaveData
    {
        5 references | Jacob Knispel, 43 days ago | 2 authors, 3 changes
        public Gameboard boardState { get; private set; }
        3 references | Jacob Knispel, 50 days ago | 1 author, 1 change
        public int difficulty { get; private set; }
        3 references | Jacob Knispel, 50 days ago | 1 author, 1 change
        public int turn { get; private set; }
        3 references | Jacob Knispel, 50 days ago | 1 author, 1 change
        public bool isSinglePlayer { get; private set; }
        3 references | Jacob Knispel, 23 days ago | 1 author, 1 change
        public int level { get; private set; }

        3 references | Jacob Knispel, 23 days ago | 1 author, 1 change
        public SaveData(Gameboard boardState, int difficulty, int turn, bool isSinglePlayer, int level)
        {
            this.boardState = boardState;
            this.difficulty = difficulty;
            this.turn = turn;
            this.isSinglePlayer = isSinglePlayer;
            this.level = level;
        }
    }
}
```

Planned Features:

Feature	Description
Overall GUI Improvements	Improvements to the style, positioning, and usability of GUI elements to allow for a better player experience.
Piece Animation	Animation of pieces moving between spaces to clarify what moves are being made.
Improved AI	Redesign of AI using Markov Decision Tree and internal board tracking known opponent pieces. Support for AI to place its own pieces.
Advanced AI	In addition to Improved AI, guesses at unknown opponent pieces and uses learning algorithms to learn, predict, and use board layout patterns.

Software Requirements Specification

This specification is meant to describe the functional and nonfunctional requirements of Stratego. These requirements demonstrate how well we managed to create a product that meets our goals and meets our prospective users needs.

Functional Requirements

As our digital Stratego game pulls its rules directly from the board game of the same name, the game must implement all of its rules in the default game. The rules of Stratego are as follows:

- The game takes place on a 10 by 10 tiled square board with a 2 by 2 obstacles in the middle two rows starting in the third column from either side of the board.
- Two players place 40 pieces on opposite sides of the board. No pieces may be placed in the middle two rows.
- The allotment of pieces for use in the game is as follows:

Piece Name	Piece Rank	# Allowed	Special Properties
Marshall	1	1	Can be captured by the Spy if attacked by it
General	2	1	
Colonel	3	2	
Major	4	3	
Captain	5	4	
Lieutenant	6	4	
Sergeant	7	4	
Miner	8	5	Can destroy bombs
Scout	9	8	Can move any distance in a straight line, unless there is a piece or obstacle in the way
Bomb	Special	6	Immovable; defeats any attacking piece except Miners
Spy	Special	1	Can defeat the Marshall if it makes the attack
Flag	Special	1	Immovable; its capture ends the game

- Each player moves one piece each turn. All pieces except scout, bomb, and flag can only move one space. No piece can move through or onto obstacles or friendly pieces.
- Moving onto an enemy piece results in an attack on that piece. Except in the cases mentioned in the above table, the piece with the higher rank is considered the winner of an attack. The loser of the conflict is removed from the board and the winner is put on the space of the defending piece. In the case of a tie, both pieces are removed from the board.
- The pieces of the enemy team are only visible during an attack.
- A player wins by capturing the enemy's flag or causing the enemy to have no valid moves.

The user will also be able to change some rules such as how certain pieces function, what the number of each piece required is, and use nonstandard pieces.

Aside from rules of the game, the users will have access to a number of other features. For example, users will have the choice of playing games as either a two-player hotseat-style match or a single-player match versus an artificial intelligence player. Additionally the user must be able to play a single-player campaign in which they play against the computer in a number of custom levels. Additionally, the user must be able to save and load game states as well as arrangements for pieces for use at the beginning of the game.

Non-Functional Requirements

We have relatively few non-functional requirements compared to our functional requirements. The most important of these are:

- Response Time - No action caused by button press should take more than a second, and no saving or loading should take more than 5 seconds.
- Reliability - The program should not crash under normal use.
- Usability - The majority of users should have no trouble determining how to do any action in the game.
- Extensibility - It should be relatively painless for future developers to make simple new features such as new pieces or campaign maps.

Software Architecture and Design Specification

This specification is meant to provide a comprehensive architectural overview of Stratego. It covers any major architectural components which compose the project. As a .NET project implemented in Visual C#, the top level (GUI) is designed and modified using a visual designer. Button presses and other user actions are handled in event methods in the main window, called StrategoWin, the first major component of our architecture. Its primary purpose is to handle GUI events and pass them onto other components and to drive the program flow. The project's "main" function merely creates and displays a StrategoWin window to the user, which will display content based on their input.

The second major architectural component is the StrategoGame class, which handles and drives game logic itself. When the user enters input indicating that they want to begin a game, the StrategoWin class creates a StrategoGame object and passes it a few parameters. This game object contains important components like the board and defines functions for handling interactions between the GUI (StrategoWin) and the game state, like the SelectPiece and nextTurn methods. More can be seen in the UML diagram in Appendix A.

The final major architectural component is that which handles the actual game logic which determines how pieces are allowed to move on the game board and in what circumstances they win, lose, or tie when fighting certain other pieces. This component is generally represented as two separate hierarchies: a GamePiece hierarchy, which is extended by classes like SpyPiece or MarshallPiece, and a BattleBehavior hierarchy, which is extended by classes like DiesToSpy which can be used to describe the behavior of a piece. These hierarchies are joined by the core of this architectural component, GamePieceFactory, which is delegated a lot of lower-level work by the StrategoGame class -- creating new piece types, changing piece type behavior, changing the default--all of which it can do dynamically.

Besides these primary components, there are some utility classes like SoundPlayerAsync, which is used to play sound files on a separate thread from the one running the GUI or the one running the game. There is also SaveLoadOperations and its sister classes SaveData and SetupData, which handle performing saving and loading, as their names suggest.

Test Strategy

For Stratego, we used NUnit, an open-source testing framework for .NET applications, to create a comprehensive test suite. NUnit allows the reuse of test methods with many different parameters to test a variety of cases with minimal code clutter. Test cases generally targeted non-GUI logic like game logic, saving and loading, and so on.

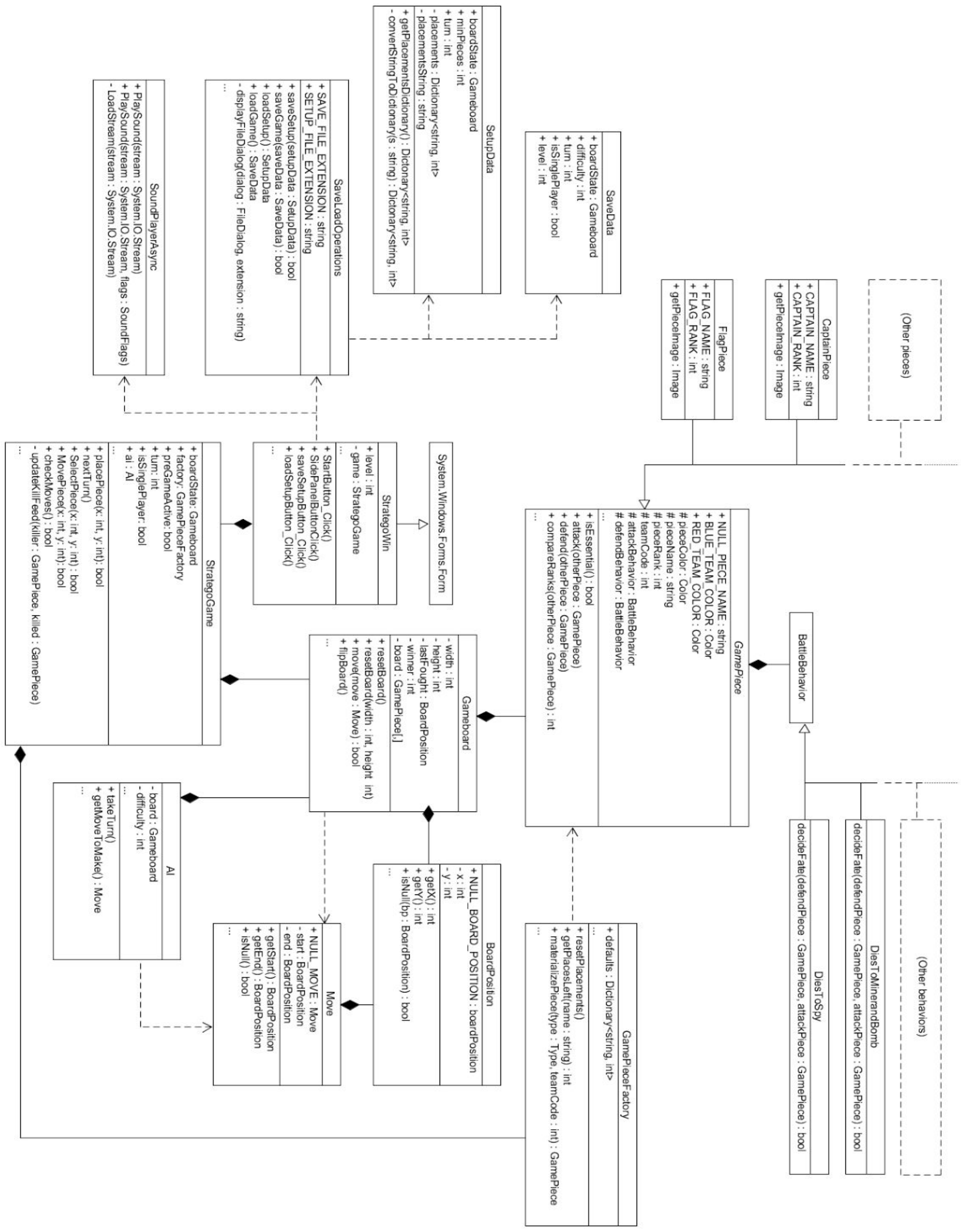
Normal testing strategy was to write a test that would call a function like “SaveButton_Click()”, which would then expect to see that the game was saved or something of the sort. This method is clearly superior to directly interacting with GUI elements because the GUI is entirely distinct from the game logic itself, not to mention how much less complicated it is. The GUI element “SaveButton” is directly linked to the method “SaveButton_Click(),” so there is no good reason to model the system actually clicking on the button when simulating it is so much easier.

Initial development of the project was test-driven, meaning that to implement a feature, tests for that feature were first put in place and then the feature implemented to meet those tests. For this reason, our test suite was very large before starting our refactoring.

A large redesign required rewriting of many of the tests, but the test suite remained mostly intact. For most small refactorings, tests were written prior to the change (if absent) in order to verify that the refactor didn’t break the code. In this way, we verified that there was no change in functionality from the original program.

Similarly, when functionality was added in the second and third milestones of this project, we added new tests to test the new functionality as would be expected. In the end, our test suite is large and healthy, and serves well to alert us when a change has subtly broken the program, which is exactly what is desirable about a test suite.

Appendix A: UML Diagram


SetupData
+ boardState : Gameboard
+ minPieces : int
+ turn : int
- placements : Dictionary<string, int>
- placementsString : string
+ getPlacementsDictionary() : Dictionary<string, int>
- convertStringToDictionary(s : string) : Dictionary<string, int>
SaveData
+ boardState : Gameboard
+ difficulty : int
+ turn : int
+ isSinglePlayer : bool
+ level : int
SoundPlayerAsync
+ PlaySound(stream : System.IO.Stream)
+ PlaySound(stream : System.IO.Stream, flags : SoundFlags)
- LoadStream(stream : System.IO.Stream)
SaveLoadOperations
+ SAVE_FILE_EXTENSION : string
+ SETUP_FILE_EXTENSION : string
+ saveSetup(setupData : SetupData) : bool
+ saveGame(saveData : SaveData) : bool
+ loadSetup() : SetupData
+ loadGame() : SaveData
- displayFileDialog(dialog : FileDialog, extension : string)
...
FlagPiece
+ FLAG_NAME : string
+ FLAG_RANK : int
+ getPieceImage : Image
CaptainPiece
+ CAPTAIN_NAME : string
+ CAPTAIN_RANK : int
+ getPieceImage : Image
(Other pieces)
System.Windows.Foms.Form
StrategoWin
+ level : int
- game : StrategoGame
...
+ StartButton_Click()
+ SidePanelButtonClick()
+ saveSetupButton_Click()
+ loadSetupButton_Click()
...
StrategoGame
+ boardState : Gameboard
+ factory: GamePieceFactory
+ preGameActive: bool
+ turn: int
+ isSinglePlayer: bool
+ ai : AI
...
+ placePiece(x: int, y: int): bool
+ nextTurn()
+ SelectPiece(x: int, y: int): bool
+ MovePiece(x: int, y: int): bool
+ checkMoves() : bool
- updateKillFeed(killer : GamePiece, killed : GamePiece)
...
GamePiece
+ NULL_PIECE_NAME : string
+ BLUE_TEAM_COLOR : Color
+ RED_TEAM_COLOR : Color
# pieceColor : Color
# pieceName : string
# pieceRank : int
# teamCode : int
# attackBehavior : BattleBehavior
# defendBehavior : BattleBehavior
...
+ isEssential() : bool
+ attack(otherPiece : GamePiece)
+ defend(otherPiece : GamePiece)
+ compareRanks(otherPiece : GamePiece) : int
...
BattleBehavior
Gameboard
- width : int
- height : int
- lastFought : BoardPosition
- winner : int
- board : GamePiece[,]
+ resetBoard()
+ resetBoard(width : int, height : int)
+ move(move : Move) : bool
+ flipBoard()
...
AI
- board : Gameboard
- difficulty : int
...
+ takeTurn()
+ getMoveToMake() : Move
...
BoardPosition
+ NULL_BOARD_POSITION : boardPosition
- x : int
- y : int
+ getX() : int
+ getY() : int
+ isNull(bp : BoardPosition) : bool
...
Move
+ NULL_MOVE : Move
- start : BoardPosition
- end : BoardPosition
+ getStart() : BoardPosition
+ getEnd() : BoardPosition
+ isNull() : bool
...
GamePieceFactory
+ defaults : Dictionary<string, int>
...
+ resetPlacements()
+ getPiecesLeft(name : string) : int
+ materializePiece(type : Type, teamCode : int) : GamePiece
...
DiesToSpy
decideFate(defendPiece : GamePiece, attackPiece : GamePiece) : bool
DiesToMinerandBomb
decideFate(defendPiece : GamePiece, attackPiece : GamePiece) : bool
(Other behaviors)