Final Project CheckPoint 1 – StarCraft Warmup
100 Points

Instructions

The template code for this assignment is posted to GitHub classroom shown on the Canvas page.

Important: When GitHub asks you for a team name, please put use the following format LastName1_FirstInitial1__LastName2__FirstInitial2, where LastName1, LastName2, FirstInitial1, and FirstInitial2 are replaced with you and your partner's actual last names and first initials. This is so we easily see who is in what group when we grade your PEX.

Overview

We will install StarCraft II, set up a project that uses Google DeepMind's pysc2 library, and write python code using pysc2 to create three bots that successfully play StarCraft against simple enemy opponents. Along the way, we'll read some documentation and do some tutorials. We estimate this will take you at least 4 hours. (1 hour for installation, and then 1 hour of focused work per part).

Installation (0 points)

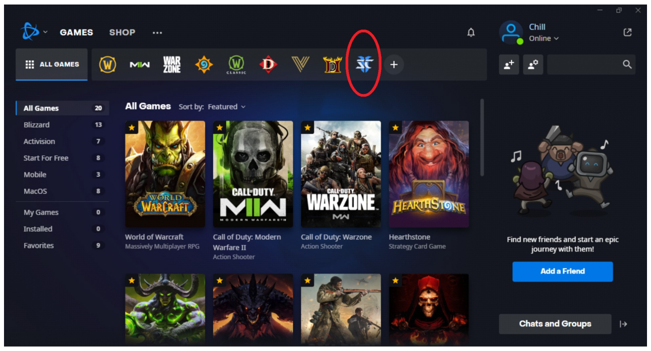
We'll be using StarCraft during the next few PEXs, so it's time to install it. This requires a few steps, so start early and be patient. We suggest doing this with your partner if possible and completing these install instructions on both computers.

Side note: MissionNet wired and wireless should both allow access to StarCraft. If wireless doesn't work and you want to use it, you can follow the instructions in this footnote[1] on your own, or with your instructor.

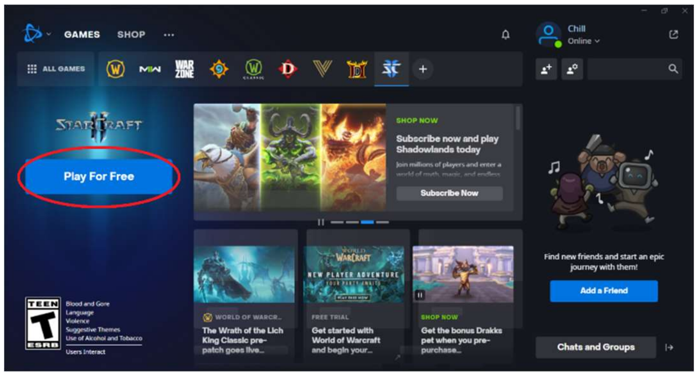
1. Install Battle.net using the installer here: https://www.blizzard.com/en-us/apps/battle.net/desktop You'll need to make an account if you don't have one. Be sure to **run the installer as Administrator**. You will also need Administrator privileges for other parts of the installation later.

[1] (1) Open a Windows command prompt, type **ipconfig** and note the wireless IPv4 Address. If it doesn't start with 10.190, you will need to forget and re-connect to it. (2) Make sure you have your MissionNet rusername and password handy. (3) Then right-click on the MissionNet connection, and choose Forget. (4) Show all available networks, and re-connect to MissionNet using your credentials. When re-connecting, it should give you a 10.190 IP address – run **ipconfig** to verify that – and therefore, starcraft should work. If it does not work as expected, try a wired connection or we can work with the MissionNet team to troubleshoot.

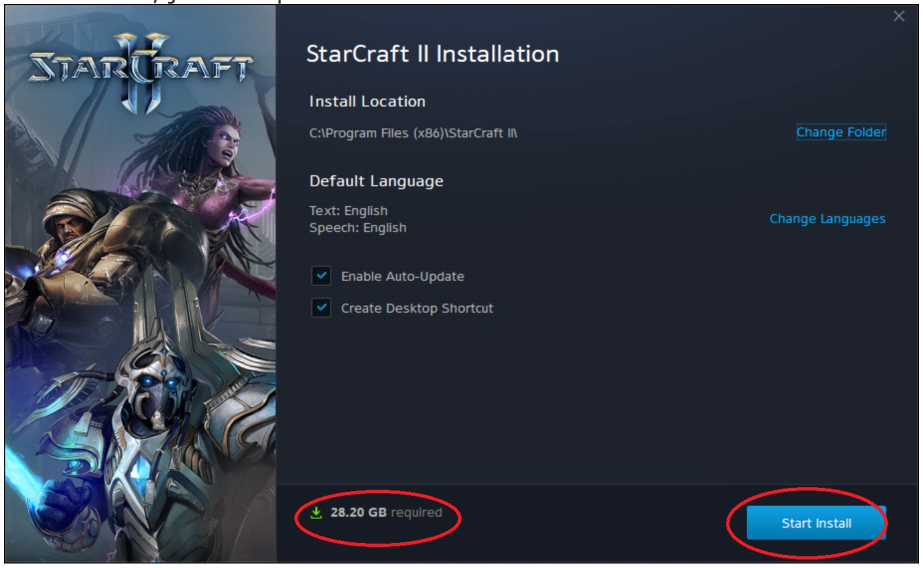
2. From the installed Battle.net app, install StarCraft II. Go to All Games.
   a. Click on the StarCraft II icon:



   b. Select Play for Free.


GAMES
SHOP
ALL GAMES
Play For Free
SHOP NOW
Subscribe now and play Shadowlands today
Subscribe Now
The Wrath of the Lich King Classic pre-patch goes live...
FREE TRIAL
Get started with World of Warcraft and begin your...
SHOP NOW
Get the bonus Drakks pet when you pre-purchase...
TEEN
Blood and Gore
Language
Violence
Suggestive Themes
Use of Alcohol and Tobacco
Users Interact
Chill
Online
Find new friends and start an epic journey with them!
Add a Friend
Chats and Groups

   c. At the next screen, just accept the defaults.



StarCraft II Installation

Install Location

C:\Program Files (x86)\StarCraft II\ Change Folder

Default Language

Text: English
Speech: English
Change Languages

✓ Enable Auto-Update

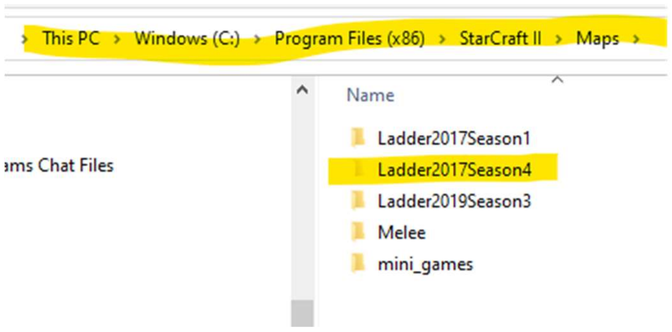
✓ Create Desktop Shortcut

28.20 GB required

Start Install

   d. Note the install size! It will take a while to install! But you should be able to do the next few steps while it is installing.

3. Download the maps we'll need to play the game
   a. Download this map: https://blzdistsc2-a.akamaihd.net/MapPacks/Ladder2017Season4.zip
   b. Unzip it – you'll get a Ladder2017Season4 folder. The password is **iagreetotheeula**
   c. Navigate to the starcraft folder (the default is C:\Program Files (x86)\Starcraft II)
   d. Create a folder called Maps in the Starcraft II folder.
   e. Copy the unzipped Ladder2017Season4 folder into the Maps folder, as shown. (Ignore the other folders there – you don't need them.)


This PC > Windows (C:) > Program Files (x86) > StarCraft II > Maps >

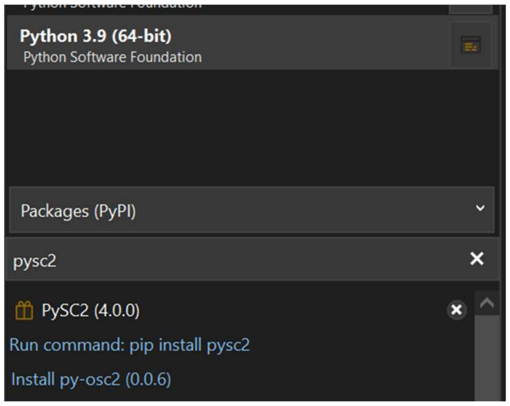
f. Repeat steps 3a – 3e using this link: https://blzdistsc2-a.akamaihd.net/MapPacks/Melee.zip
g. Many more maps at [Blizzard]() and more explanation [at deepmind]() (both optional).

4. (Optional) At this point, you could spin up StarCraft II to get some background. TL;DR: there are three races: **Terran**, **Protoss**, and **Zerg**, who are battling each other.

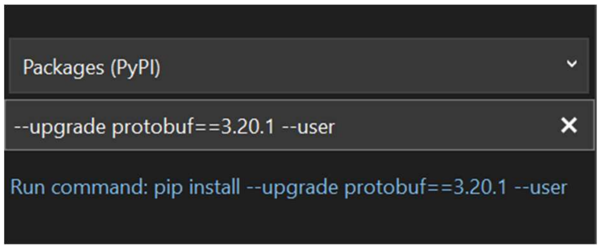
5. Connect to the github classroom (link above) and check out the starting code. Clone the template and open it in Visual Studio Community 2022.

6. These next steps show how to download DeepMind's pysc2 (**PY**thon**S**tar**C**raft**2**) library using their [github page](), but are customized to Visual Studio:

   a. Go to Python Environments on the right and choose the Python 3.9 that came with the Visual Studio install If you don't see Python Environments, use View > Other windows > Python environment. (**Note**: as of Nov 2022, we've had the most luck with python 3.9. One cadet had success with 3.7, but not on 3.11. If you need python 3.9, go to https://www.python.org/downloads/ and download.)
   b. Select the environment. Then below the environment name, look for Packages (PyPI) – click the dropdown to select it if it isn't shown. Then in the search window that says "Search PyPI and install packages", type **pysc2** and then below that, press **Run command: pip install pysc2**. It will take a few minutes to install this package and its dependencies.



   c. **IMPORTANT:** There is an issue with the protobuf library version that pysc2 uses. If you were to run it now, you'd get an error: [python - TypeError: Descriptors cannot not be created directly - Stack Overflow]() To downgrade protobuf and avoid this issue, type `--upgrade protobuf==3.20.1 --user` in the same window you typed pysc2 and then press the Run command: ([reference]()). If this doesn't work, complaining about virtualenv, then try it again without the **`--user`** at the end, and if it still gives an error, try re-starting Visual Studio.



```
--upgrade protobuf==3.20.1 --user
Run command: pip install --upgrade protobuf==3.20.1 --user
```

d. The library is now installed. The default location will be something like C:\Users\Your.Name\AppData\Local\Programs\Python\Python37\Lib\site-packages\pysc2 This is handy in case you need to look at source code.

7. Test your install by running **`empty_agent.py`**.

Question 1 (15 points): Learn the Basics about StarCraft and the pysc2 Library

(We expect this to take ~1 hour.) Steven Brown has created a series of tutorials (list on medium) about pysc2. Go to his tutorial (Build a Zerg Bot with PySC2 2.0). Read it slowly and carefully to learn about the basic game loop and how to build a basic, hardcoded agent. Take some notes – it will help with the deliverable for Q1 (described below). Build the code in the provided file **`build_zerg_agent.py`**. He has a link at the bottom of the tutorial to the finished code in github. It's up to you if you want to type or copy the code in step-by-step or just grab the whole chunk, but the goal here is to understand what he is doing. You may find this table helpful. You'll see each in the code.

Concept \ Race:	Zerg	Protoss	Terran
Initial base:	**Hatchery**		
Basic worker unit:	**Drone**		
Source of supply (food):	**Overload**		
Basic fighting unit	**Zergling**		
Building that creates the fighters:	**Spawning Pool**		

If you know nothing about StarCraft, you'll also find liquipedia.net helpful – it's a well-organized encyclopedia of StarCraft knowledge. This entry on Zergs gives an overview, and then click Zerg Units to see the buildings (in a tree) and the units that each building can build (lists underneath the tree).

Once you have completed the code, put a breakpoint next to the first line in the `ZergAgent`'s step method (the line that starts with `super`) and debug the program. When it pauses at the breakpoint, look at the variables in the Locals window (Debug > Windows > Locals). Expand the `obs` object, then `observation` within it. It has many items in it. Note any of the variables that start with the letter **m**. (Full disclosure: our goal isn't so much for you to find a specific item as it is for you to launch the debugger since it will be useful while you are writing your own code later.) `Observation` is a large data structure that contains almost all of the information occurring in the game at the current timestep. You will use this information to build an intelligent agent later in the project.

Once you are done the tutorial, write a summary of what you learned in the comments at the top of the file. Also include your object starting with "m" from the previous step.

Commit and push your code.

Question 2 (15 points): Learn Raw Functions

Note: it's usually helpful to change *drivers* (which partner is typing) between questions. So your partner should pull the code from question 1 from the repo, and then start question 2.

The previous tutorial gave you a decent start to orienting yourself on pysc2, but the code presented suffers from two limitations:

1. Your bot can only see what is close to it (on the Screen window), which is a small part of the whole world (the Minimap window).
2. Each unit takes two steps to act: first select it, then do the action.

Both of these limitations are overcome using a paradigm that is designed for StarCraft bots: RAW_FUNCTIONS. Steven Brown has written a second tutorial about this (Create a Protoss Bot Using Raw Observations and Actions in PySC2). Read and do this tutorial. **Skip step 1** about cloning pysc2 dev branch. Add the code to **`build_protoss_agent.py`**. Spoiler alert: he is building the same bots as the previous tutorial, but using the Protoss race. While there are some differences, it is all pretty analogous. Do you agree with this table?

Concept \ Race:	Zerg	Protoss	Terran
Initial base:	Hatchery	**Nexus**	
Basic worker unit:	Drone	**Probe**	
Source of supply (food):	Overload	**Pylon**	
Basic fighting unit	Zergling	**Zealot**	
Building that creates the fighters:	Spawning Pool	**Gateway**	

Similar to the last tutorial, please write a summary of what you learned, in the comments at the top of this file.

Reminder: commit and push to the repo, then partner pulls.

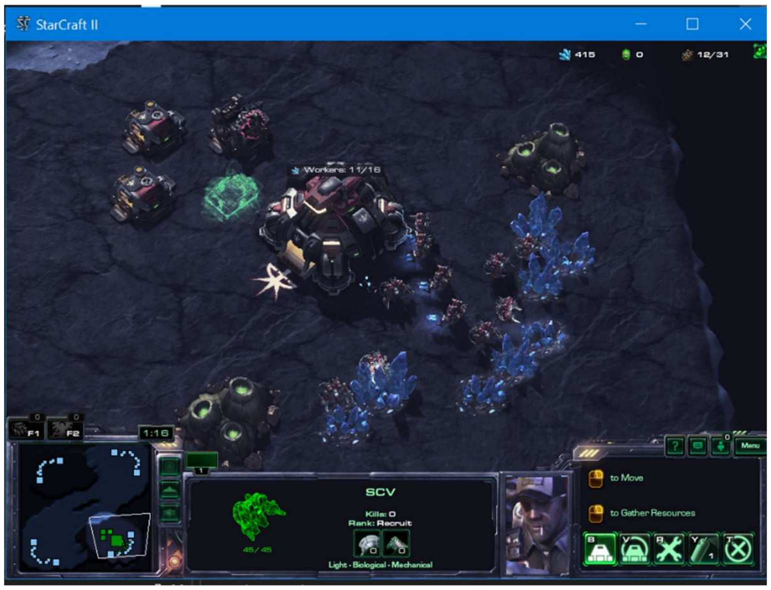
Question 3 (70 points): Build Marines and Win Battles Against Wimps

For the rest of the final project, we will build up a friendly base and then train an artificial neural network to defeat an adversary. We will use a simple scenario (the Simple64 map) as we will not have enough time in the remainder of the course to train a neural network to play an advanced adversary. You will also set the adversary difficulty to `very_easy`. To get started, you will build your friendly base with a series of scripted actions:

In **`build_marines_agent.py`**, write code using RAW_FUNCTIONS to do the following:

- Build 4 supply depots (Define this as a variable, near the top of your program that you can easily modify).
- Build 2 barracks (Define this as a variable, near the top of your program that you can easily modify).
- Train as many marines as your supply allows, using both barracks, and send them to attack the enemy on the opposite quadrant of the screen.

You are only required to play as the Terran race. You will also notice that your base will always start in the top-left or the bottom right corner of the Simple64 map (the game randomly chooses which). The enemy location will always be in the opposite corner to your friendly base (for example, if your base starts in the bottom-right corner of the map, the enemy base will be located in the top-left corner).



The Simple64 map is named based on its coordinate system. There are 64 locations in the x direction and 64 locations in the y direction. Use this information when determining the coordinate locations of where to send units. NOTE: The entire 64 locations are not available. Experiment with sending units to different locations to determine where the boundaries are for the accessible map. In the game shown here, the 64x64 minimap (bottom left) shows that the highlighted screen, where the Terran base is, is in the bottom-right corner of the world. Thus, the enemy will be in the top-left corner.

As you'll read in the specifications for checkpoint 2, your code will be generalized to allow movement to any of the four quadrants. This allows for more interesting game play while keeping the learning problem for the neural network simple. The quadrant will be chosen randomly while exploring and chosen by the neural network once it has been trained on game data.

Hints:

- You may assume the same situation as in the Protoss bot tutorial: that your base will be on one corner and the enemy will appear in the other corner.
- To build multiple bases, you'll need to vary the location in which each is built. The randomization pattern that generates x_offset and y_offset values and used in the Protoss bot attack location code should work nicely here. Use `random.random()` to generate a random number from 0.0 to 1.0 and multiple this by your offset to generate a random location each time. Just keep issuing build actions until you have 4 depots. Ditto for the 2 barracks.
- You can alternate barracks used to train marines by keeping track with a variable, or (and perhaps easier), whenever you issue a command to train a Marine, pick 1 of the 2 barracks at random.
- You may notice that you issue an action command and the game does not perform any action. This is usually caused by the game's inability to execute that action. For example, if I issue the command to train a marine, there may not be enough minerals available to build the marine. Use the debugger and liquipedia.net when you are trying to determine why an action does not execute.
- You may find it helpful to go back to https://liquipedia.net/starcraft2 and search for Terran to learn the relationship between the units. (I used that info to complete the table above – it helped organize my thoughts.)
- If you need specific names of functions, go to https://github.com/deepmind/pysc2/tree/master/pysc2/lib, open actions.py and search for RAW_FUNCTIONS. It will give the whole list you need.

Grading Rubric

Expected Functionality	Points Allowed	Points Earned
(Question 1) Describe what your team learned from Steven Brown's Build a Zergbot with PySC2 tutorial in the comments sections of the `build_zerg_agent.py` file. This should be roughly 1 paragraph long at whatever level of detail your team finds helpful.	12	
(Question 1) The comments in that file also include the environment objects that start with m, as requested.	3	
(Question 2) Summarize what you learned about RAW_FUNCTIONS in the Create a Protoss Bot tutorial. Add comments to `build_protoss_agent.py` file.	15	
(Question 3) Create a function that performs scripted actions at the beginning of a StarCraft II game on the Simple 64 map (70 points broken down below). Write code in `build_marine_agent.py` file.		
All code (except environment startup actions) is contained within a function	5	
All actions are performed using RAW_FUNCTIONS	5	
No hardcoded magic numbers (such as the number of supply depots of barracks to build); instead saved in constants near the top of the function.	5	
Scripted game is contained within an epoch loop. In other words, after the player has won or lost, another StarCraft II episode will immediately start.	15	
4 Supply depots and 2 barracks are built as soon as resources are available. You may define static locations for these buildings or randomize the location each time the game is run.	15	
Barracks begins to build marines as soon as it is completed	5	
Once marines are produced, begin moving marines to the	20	

opposite quadrant than the marines started in. See hints on how to accomplish this.		
Total	**100**	

Other references:
https://gamescapad.es/building-bots-in-starcraft-2-for-psychologists/ (good introduction to building StarCraft bots for beginners)
https://github.com/deepmind/pysc2/blob/master/docs/environment.md (basic description of the StarCraft Environment object)