BSC – HGP - Project
Go Specification

1. Assignment Information

Course	BSCO / BSCH
Stage/Year:	3
Module	HCI & GUI Programming
Semester	1
Assignment:	3 – Project
Date of Issue:	21/11/2019
Assignment Deadline:	23/12/2019
Assignment Weighting:	60% of Module

2. Introduction

Important: Please Read

Please read the project description in full before you start coding. Failure to do so will result in you missing out on important information and reducing your ability to avail of marks. Please attempt to work in line with the file structure provided. Any significant departure from this template may result in loss of marks

In this project, you will be tasked with building a fully working game of Go.

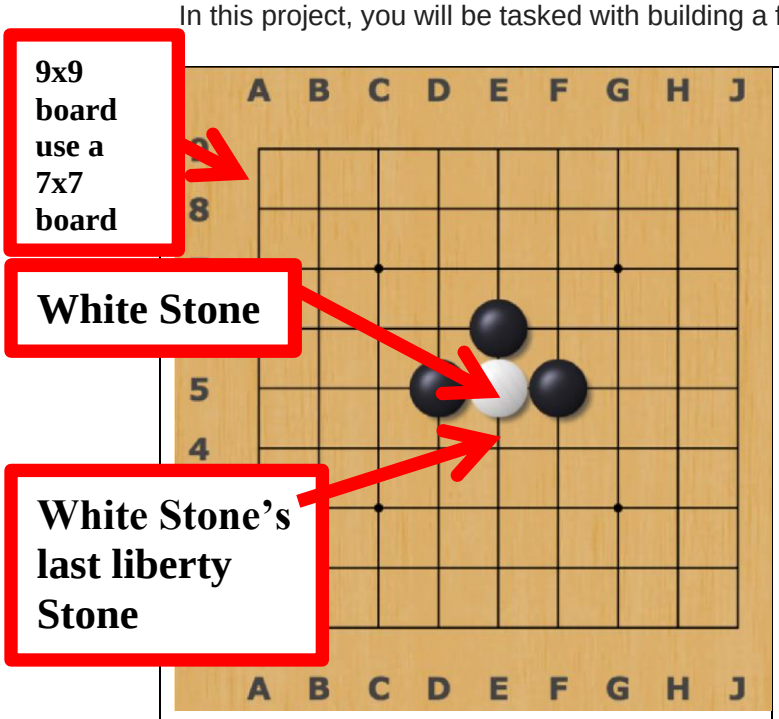


Figure 1. White can be captured by black placing a stone on position E4.

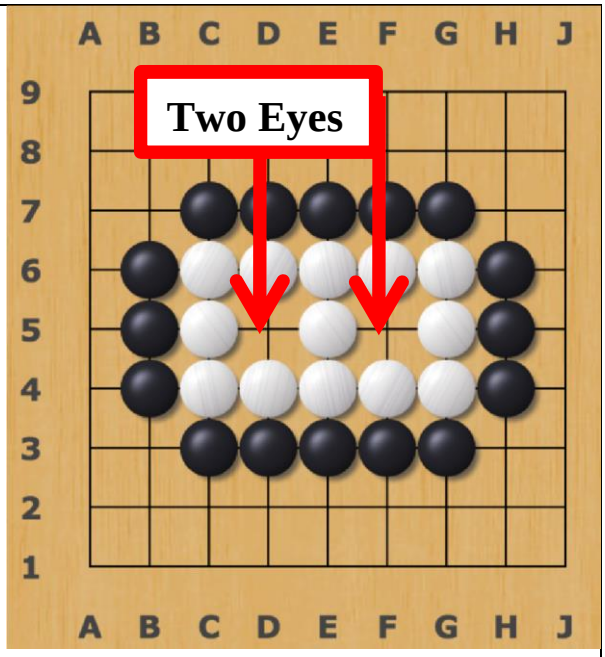


Figure 2. This white string is safe as it has 2 eyes.

Go ("encircling game") is an abstract strategy board game for two players, in which the aim is to surround more territory than the opponent.

3. Explanation of the Game

The game was invented in China over 3,000 years ago and is therefore believed to be the oldest board game continuously played today. It was considered one of the four essential arts of the cultured aristocratic Chinese scholars in antiquity. Despite its relatively simple rules, Go is very complex, even more so than chess.

Computers have only recently been capable of beating human masters. Have a look at the following for more details: https://deepmind.com/research/alphago/.

3.1 Initial board layout

We will use a 7x7 board to ensure quick game play and reduced complexity. Go is commonly played on a 13x13 and 19x19 grid. Black goes first. Stones are placed on the grid intersections.

3.2 Movement

Black plays first, with black and white taking turns. A *stone* can be placed at any unoccupied intersection of the board with limited exceptions.

1. **Suicide rule:**
   You cannot place a stone which will immediately have no liberties. https://youtu.be/JWdgqV-8yVg?t=9m00s
2. **KO Rule (Eternity Rule):**
   Previous game states are not allowed. Keep a list of previous game states which must be checked before stones are placed https://youtu.be/JWdgqV-8yVg?t=7m35s

3.3 Determining a Winner

When a player thinks their territories are all safe, and they cannot gain any more territory, reduce their opponent's territory or capture more strings, instead of playing a stone on the board they pass and hand a stone to your opponent as a prisoner. Two consecutive passes terminates the game.

3.3.1 Awarding of Points

- stones captured
- territory controlled by a colour

3.3.2 Additional Rules and Information

- A detailed set of rules is available here https://www.britgo.org/intro/intro2.html.
- A cartoon tutorial is available at https://www.britgo.org/cartoons/index.html .
- A well-structured version of the rules is available here https://en.wikipedia.org/wiki/Rules_of_Go
- An extensive list of GO terms available at https://en.wikipedia.org/wiki/List_of_Go_terms . You are not required to know these to complete the project, but they will expand your awareness of the game.
- There is a lot of additional information on Go some of it code related at https://senseis.xmp.net/

3.3.3 Interesting situations

- Seki (Impasse): A board position may arise where a play can capture opponents' piece, but the opponent can immediately recapture a string of pieces. Whoever goes first loses. https://youtu.be/JWdgqV-8yVg?t=11m12s
- Having eyes is a strong position

3.3.4 Handicaps

• You may implement this as an advanced task, but you will need to do more research. Typically, white gets 7.5 points for going 2nd. The .5 point is to avoid a tie.

Features(low-level), Marks & Penalties

The required features are listed here in detail. Failure to implement a feature will result in loss of marks. There is a degree of flexibility in the method of implementing these features. If you are unclear whether or not your proposed method of implementation is acceptable, please ask the lecturer. Ensure that you attempt all components.

Section	Subsection	Task #	Marks	Details
Application (80%)	Board	1	15	Generate the basic board for your application. It should display a full Go board of side size 7.
	Menus/ Button/ Labels	2	20 (6 x 3.3)	Add code and menus/buttons/labels to your application to a) Show how to play your game including rules b) Show how many prisoners each player has taken c) Show how much territory is controlled by a player d) Show whose turn it is e) Allow player to pass f) Allow the game to be reset
	Placement	3	10	Implement placement of stones using mouse clicks.
		4	10	Implement placement of stones in valid locations only – suicide rule
		5	10	Implement placement of stones in valid locations only – KO rule
		6	5	Implement capture of stones – single stone
		7	5	Implement capture of stones – multiple stone
	Winner	8	10	Implement winner detection, the game should then end immediately with an appropriate notification. Two passes.
	Additional Feature	9	15	Select one of the following: - 2 timers 1 for each player to implement speed Go. Each player should have 2 minutes to make moves. The 1st players timer should start to count down when the game is started, 2nd players timer counts down when 1st player has completed his move and so on. If a player runs out of time then they will lose the game. - Animation of moves - Implement a handicapping system - Other additional feature of your choice with similar complexity
Documentation	Code Documentation	9	10	Clearly Comment Code in file - Explanation of method functionality, data structures and underlying logic - Explanation of parameters of methods Kept is precise and clear, complete for all code elements, review provided links for additional tips
	UI Design Description	10	10	Use template provided. Include screen shots, write clearly under all headings, explain all choices no matter how basic
The following tasks are only required when the group size is 3				
		11	15	The ability to undo/redo moves
		12	10	Animation of pieces (e.g. pieces grows/spins/flashes) under the following circumstances a) A piece is places on the board b) Pieces are captured c) A winner is determined
			125	**Total Marks**
				If group size =2 then 100 marks =100%
				If group size = 3 then 125 marks = 100%

Deduction	Error & Reason
-30	Non-executable code submitted. Encourages student to build robust code. Reduces marking time.
-20	Non-standard libraries used, only standard SDK allowed. Ensures equal workload of all students. Reduced marking time by avoiding the installation of custom libraries for specific submissions.
-10	Wrong compressed file format (zip and rar are accepted). Encourages student to distribute resources in easy to read formats. Reduces marking time as additional decompression apps do not need to be installed.
-10	Wrong folder structure (see project introduction) . Encourages students to present work in a well structured format. Reduces marking time to determine location and presence of component
	deductions for bugs
	standard late deductions

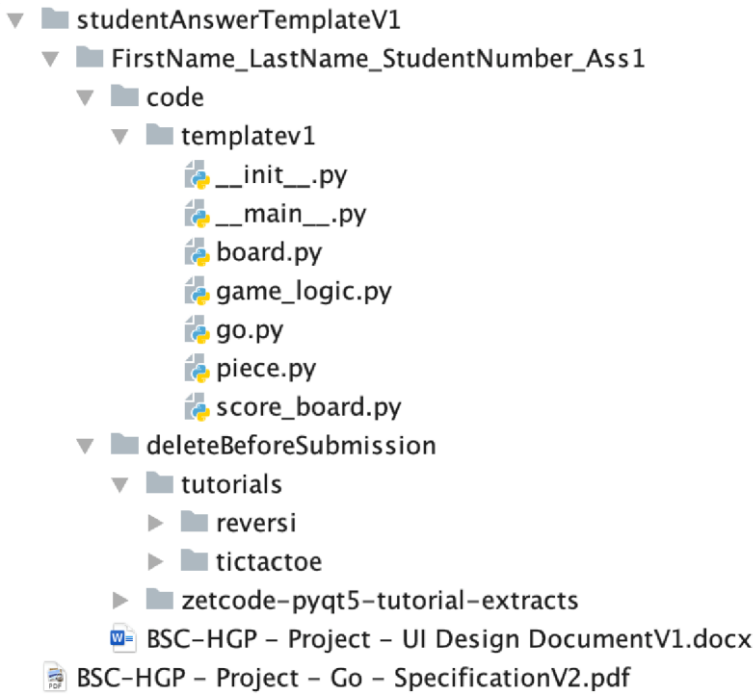
Table 1. Marks Allocation

Each feature is awarded marks based on

1. **Present:** if the feature is present in the application
2. **Function:** if the feature contributes to a well working app, higher marks will be awarded for customization of the function or attributes of the widget
3. **Well Designed:** if the feature is incorporated well into the application obeying GUI design principles.

Resources to Assist You

studentAnswerTemplate is available on Moodle to download. It contains the following folder and files



- *code* - **edit/add** files in this folder to complete you code solution
  - *__main__.py* is the file to be run to execute the project
- *deleteBeforeSubmission* - **do not edit** any files in this folder ○ *reverse – python non-gui version of the game with web reference to tutorial* ○ tictactoe – *python non-gui version of the game with web reference to tutorial* ○ zetcode-pyqt5-tutorial-extracts
  - *all* – contains all the examples
- *BSC-HGP – Project - UI Design DocumentV1.docx* – **edit** to explain design choices and highlight additional features, and illustrate what is working and not working. • *BSC–HGP - Project - Specification.docx* – **do not edit** this document

6. Submission

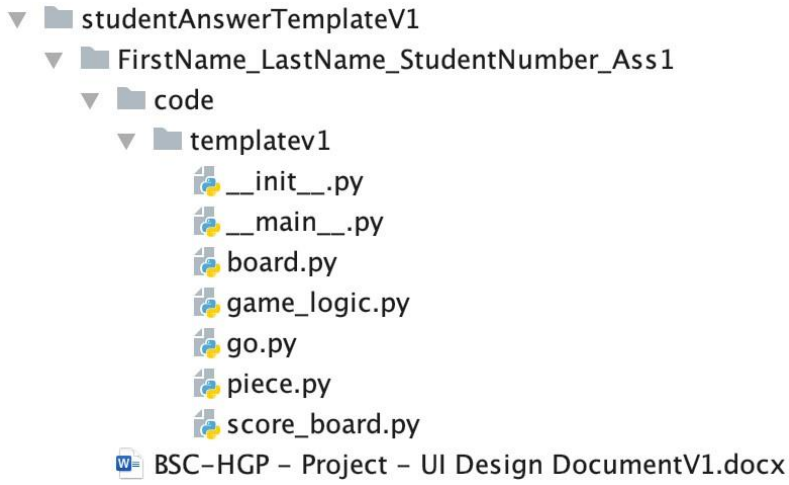
6.1 Identical Submissions

The final submission by both partners should be identical. This can be ensured by following these steps:

1. Contact your partner and decide whose version will be the final version.
2. Partner A will be the partner in possession of the final version.
3. Partner A zips and upload the final version.
4. Partner A emails this zip to Partner B
5. Partner B uploads this zip. B can unzip the file to look at it but uploads the original zip (not unzip and rezip).

6.2 Submission structure

Your final submission should be structured as below



- **Rename** FirstName1_LastName1_StudentNumber1_StudentNumber1_ FirstName2_LastName2_StudentNumber2_Project to your details
- **Compression** folder to zip or rar
- **Submit** to Moodle

7. Project Plan

Below is a suggested plan to assist you in completing this project.

Time Frame	Description	Outcome
Day 1	**Learn the Game** 1. View "How to play go in 2 mins" https://www.youtube.com/watch?v=Jq5SObMdV3o 2. View "How to play go in 15 mins" https://www.youtube.com/watch?v=JWdgqV-8yVg 3. Read the rules of Go https://www.britgo.org/intro/intro2.html 4. Read the online cartoon tutorial https://www.britgo.org/cartoons/index.html 5. Play a game of go online https://www.cosumi.net/en/ 6. Play go with your partner http://ba.net/juegos/go/#blank-9 7. Discuss your understanding of Go with your team. 8. Discuss your understanding of Go with another team.	In depth knowledge of game
Week 1	**Hack Your Code** 1. Start working on the provided template 2. Share this using GitHub (preferred) or google drive etc. 3. Start a work log of how the work is to be divided between partners – a google drive document would be good for this. 4. One member could work on adapting the appearance of the GUI to mimic GO while the other researches the game logic that will be required to implement GO. 5. Add in the GUI requirements. You can connect them up later. 6. Discuss your progress with your teammate and other teams. **N.B.** Make sure to test your application after each small change.	At the end of this phase your GUI should LOOK LIKE a game but most of the functionality will not be present. You should be able to place stones on the board.
Week 2	**Workout the Game Logic** This is the most complex part of your project. You should have a good idea on paper about how this game should be played before you implement this in GameLogic.py 1. Discuss game logic with you group 2. Discuss game logic with other groups. 3. Agree on what methods should be written, what they will do and what they will return. 4. Compare these methods on a logical level with other teams so you know you are on the right track. DO NOT SHARE CODE with other groups.	

Week 2	**N.B.** If you have spent time working out the game logic on paper/in written documentation, then the next part will run more smoothly. If you attempt to hack your code to get it working, you will fail in the next step. You will not accidently write the correct code … planning is essential.	Agreed set of methods.
Week 3	**1.** For each of the methods that you have defined generate detailed comments in GameLogic.java prior to writing the methods. **2.** Define tests which you can run to determine if the code is performing correctly. This testing should be detailed and performed often. When you have a small code base errors are easy to find. As your codebase becomes larger it becomes more and more difficult to locate and correct errors. Below are 2 examples of how you can could test: **a.** Formal Test Classes if you have done this previously **b.** Text output to the console or visual feedback from you GUI.	You should have a game that can be played but contains several errors (large or small) you should have some code in place to help you figure out where specifically the errors are located.
Week 4	You will be focusing on generating a working game. You will have several errors and you will be figuring out if they are logical errors (i.e. you made a mistake in the work in Week 2) or if they are implementation bugs (i.e. you made an error in Week 3).	Reduced error count.
Week 5	**1.** Ask other students to test you game to see which features are present and which are not. **2.** Work on adding the advanced features. Ensure that it is easy to roll back the code to previous states.	Fully working debugged code.

8. Working in Pairs

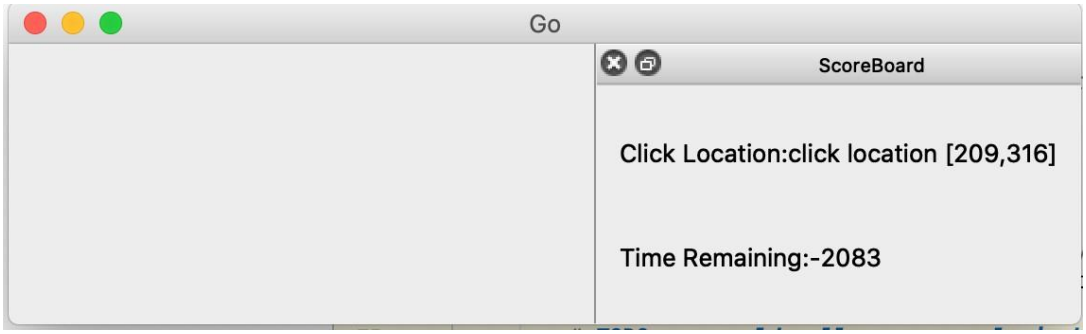
- Agree on a common IDE e.g. PyCharm for code development. Converting from one project format to the other is messy at best.
- If you can use a SCM (source code manager) to develop and share code, please do. I would recommend using GitHub if you can (particularly if you have access to a git server somewhere where all changes can be uploaded).
- Decide on a development strategy. Try to work on independent parts if you can. Working in parallel will cut the time in half.

- "Weeks of programming can save you hours of planning" a quote from Scott Meyers. Basically, it means design the project first before implementing it. This means deciding on your data structures, UI design, view design etc. before you start. Pay attention to your data structure for the game board and how you implement the rules.

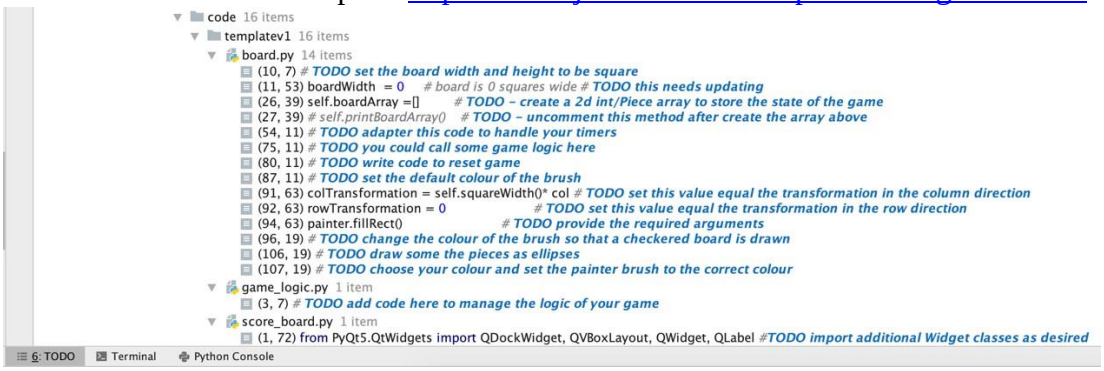
9. Steps to Complete Assignment

Below are a couple of steps to get you started.

1. Start with the template and run *__main__.py*



2. Address ToDo items in template https://www.jetbrains.com/help/idea/using-todo.html



3. Pay close attention to the painting tutorials http://zetcode.com/gui/pyqt5/painting/
4. Print regularly to monitor your progress. You could also consider logging.
5. Add intuitive widgets https://doc.qt.io/qt-5/gallery.html

10. Documentation

1. Qt Documentation
    1. Widgets: https://doc.qt.io/qt-5/qwidget.html
    2. Modules: https://doc.qt.io/qt-5/qtmodules.html
2. PyQt Documentation
    1. https://www.riverbankcomputing.com/static/Docs/PyQt5/api/qtwidgets/qtwidgets-module.html
3. Documenting Your code
    1. https://realpython.com/documenting-python-code/ (you can just use # and a good explanation!)