**2. Language Tutorial**

In the tutorial, we will walk you through creating your first GBL program.

Users could refer to our program “initial.gbl”, which provide the user an architect of a board game they may use.

We divide a game into five parts:

(1) Initialize: the initial value of a game.

(2) Win condition: in what condition which player will win.

(3) Is the sprite legal: whether the operation of a player makes is right?

(4) Update the game: when the player place a sprite, what will change in the board.

(5) AI (optional): users could write the AI of themselves.

Thus, the initial GBL file will include the content as “initial.gbl”.

Then we move to how to create a GBL program.

**Step 1: Creating your .gbl sourse file**

Create a file ends with .gbl.

**Step 2: Add Game information.**

Users can refer to “initial.gbl” and add the five parts in the .gbl file. There are three build-in classes named Game, AI and Main. Users should create their class extend from these three classes first, then add Initialize information in Main-child class, add Win condition, Islegal, Update information in Game-child class, add AI in AI-child class. We explain every parts as follows:

2.1 Initialize. In this part, users may write the initial information of the game, e.g. the size of map, the number of players, the name of players, if user fight with an AI and so on.

2.2 Win condition: In this part, users may add the win condition, give the information saying that in what condition, which player will win. For example, in game Gobang, the sprites of any player give a line, of which the number of sprites is larger than or equal to five, then the player win. The return of Win function is the Winner ID.

2.3 Islegal: In this part, users may give the result whether the operation of a player makes is right. For example, in game Gobang, a legal operation is that a player put a sprite in a vacant grid. The return of Islegal function is a bool variable indicating whether it is legal.

2.4 Update: In this part, users may give the information that when a player make a right operation, what will happened in the board. For example, in game Gobang, the Update is that a sprite is placed in a vacant grid. But in other games, some sprites may be delete from the board. The return of Update function is the coordinates of sprites which will be deleted.

2.5 AI: In this part, users could write their smart AI to fight with players. The return of AI function is the sprite position AI give.

**Step 3: Create .py file.**

Run: $cat <your file name>.gbl | ./microc.native > gbloutput.py

**Step 4: Run gbloutput.py**

Run: $python gbloutput.py

**Step 5: Play your game!**

The player could choose where they want to put a sprite by click on a grid. Enjoy!

**6. Text Plan: Explain why and how these test cases were chosen**

**7. Lessons Learned**

**Each team member should explain his or her most important learning**. **Include any advice the team has for future teams**

In the course, what the most important I learn is not the Ocamel language, but a thinking about compiling. PLT and also the project tell us how a computer understand the language our human use, like C/C++, python or some other programming language. The course give me a new view to see our language. I can see in some day, our computer could fully understand I am talking now. As for the advice, I think the team could start the project early and make a clear goal they want to realize.