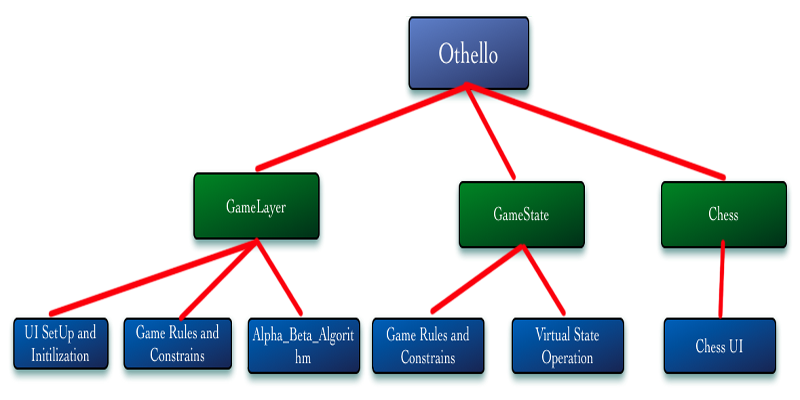
**Othello Document**

AppDelegate class and RootViewController class are two default-generated classes used to set and switch screens.

GameConfig class is also a default-generated classes used to set definite game settings

This Document will only focus on the GameLayer class, GameState class and Chess class which are self created.



1. **GameLayer**

GameLayer class contains the main game layer and strategy of the game. It can be divided into 3 parts – the Scene set-up, the implementation of the game rule and the Alpha\_Beta\_Search algorithm with minimax .

* 1. The Scene Set-up

+(CCScene \*) scene

-(id) init

These two method create the scene and set up all the user interface including menus, score labels and background and initialize the initial state.I also create a rectangle array to divide the game board into 16 pieces so each time human player touch location ,it will trigger the event in that retangle position base on the state of that square (empty , can touch or is filled).

* 1. The Othello rule

-(void)CheckPutChess;

-(void)Checkswollow;

-(void)CheckScore;

-(NSMutableArray \*)getHumanActionList;

-(void)clearPutPostion;

These four method determine the constraints and rules of the game. CheckPutChess check if an empty space can be put and draw a dot in that space and getHumanActionList will return a array of the can-put space.

Checkswollow check when a chess is put which chesses will be sollowed to reverse color and CheckScore keep tracks of the current number of white chess and black chess and when there are no valid movement for each side, it will announce a winner of a tie base on the scores.

* 1. The Alpha\_Beta\_Search algorithm

-(int) Max\_Value:(GameState\*)state a:(int)alpha b:(int) beta;

-(int) Min\_Value:(GameState\*)state a:(int)alpha b:(int) beta;

-(void) Alpha\_Beta\_Search:(GameState\*)state;

-(BOOL) cutOnDifficulty;

Alpha\_Beta\_Search, Max\_Value and Min\_Value implement the key Alpha\_Beta\_Search search base on the algorithm in the book.In order to calculate the optimal movement in the backend, I create another class GameState to store the virtual state information data. The search algorithm take a GameState object ,copy the current state to the virtual state and call Max and Min methods with pruning to determine the next move.

In the Code:

if([[valueList objectAtIndex:j] utility] == v){

//state.actionList[i];

saveValue[x][y]=-1;

[Chessarray[x][y] setChessType:saveValue[x][y]];

current\_x=x;

current\_y=y;

NSLog(@"AI PUT");

[Nowstate release];

return ;

}

CutOnDifficulty is the fuction set the difficulty of the game. It cuts the tree of certain depth for a easier AI in 2 and 5. Since the Depth won’t reach 20, when cut depth is set to 20, it will generate the whole tree.

1. **GameState**

GameState is a virtual state used to store all the state information only for the Alpha\_Beta\_Search. It mainly contains a 4x4 array with value 1 for black, -1 for white ,0 for blank, 4 for a position that black chess can be put and -4 for a position that white chess can be put.

-(NSMutableArray \*)getActionList;

-(GameState \*)getNextState: (int)x: (int)y;

-(BOOL)testTerminate;

-(int)getStateValue;

-(void)CheckAiswollow;

-(void)clearActionList;

These methods implement the same game rule similar to those in the GameLayer class.

-(int)returnArrayValue:(int)x:(int)y;

-(void)setArrayValue:(int)x:(int)y:(int)val;

-(void)SetNextState:(int)x:(int)y;

-(void)Copytostate:(GameState \*)state :(GameState\*)tostate;

-(BOOL)isSameState:(GameState \*)state :(GameState\*)tostate;

These methods implement the basic operation on the virtual states.

1. **Chess**

This class implement the UI for each Chess.

-(id) initWithPosition:(CGPoint)pos theGame:(GameLayer\*) game ;

This method Initialize a chess.

-(void)setChessType:(int)ballT;

This method decide which image should be attached to a chess. In Default case, All the 4x4 square are set to transparent images.

1. **Screenshots**



Game Start!



Game Playing!



Game Ends

::Screen shot 2011-12-07 at 下午06.31.34.png

Output Nodes number , Depth number and pruning number in the log.