Project Sprint #4
John Chirpich

Github link: https://github.com/johniscool1/cs-449-project

1. Demonstration

Youtube/Panotopo link: Link

2. User Storie for the Computer Opponent Requirements

ID	User Story	User story Description	Priority	Est Time
8	Computer Component	As a player, I want to choose a computer player so that I can test my skills.	1	3

3. User Stories for the Computer Opponent Requirements

User Story and Name	AC ID	Description of AC	Status
8 Computer Player	**8.1**	**AC 8.1 Computer Player is selected Given the player selects for a CPU player to play when they select the CPU as Player 1 or 2 or both Then Have the cpu play the correct player**	**done**
	8.2	**AC 8.2 Computer Player is "Compitent" Given the player is one letter off of an SOS sequence When its the CPUs turn Then I want the CPU try to score**	**done**
	8.3	**AC 8.3 Computer VS Computer Given The player has selected CPU vs CPU When The player starts the game Then the Computer should play its self**	**done**

4. Sumamry of Source Code

Source Code file name	Production or testcode?	# of lines	
main.cpp	pro	16	
screen_def.hpp	pro	133	
screen_def.cpp	pro	498	
game_logic.hpp	pro	138	
game_logic.cpp	pro	1177	
unit_tests.cpp	test	323	
	total	2285	

5. Production Code vs New User stories/Acceptance Criteria

User Story ID & name	AC ID	Class Name(s)	method Name(s)	Status	Notes
8. Computer Oponent	8.1-3	GameLogic, CPUlogic (public PlayerLogic)	GameLogic::CPUseek, GameLogic::CheckIfScore, GameLogic::FindButton	Done	

6. Tests vs New User stories/Acceptance Criteria

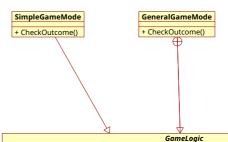| User Story ID and Name	AC ID	Method	Description(expected I/O)
8. Computer oponent	8.1&2	TEST_CASE("8.1 & 8.2 CPU player is compitent")	Creates a board, then has player 1 create an SO, then has the computer player play to see if the Computer player is compitent.
	8.3	TEST_CASE("8.3 CPU v CPU")	Creates a gameboard and has the computer play its self in a general game. Then checks if all 100 spaces were played (10x10 board) and then chackes if the computer played on a space already played.

7. Present the class diagram of your production code and describe how the class hierarchy in your design deals with the computer opponent requirements

For class Hierarchy relating too the commputer opponenet, I create a class CPUlogic woth the parent class playerlogic. The Computer player is a player so it is a child of the parent class player logic. Due to all the data needed to use the cpu player being contained in the gamelogic class, all the logic pertaining to the CPU player is in the game logic class.

CPULogic

+ CPUplayer : int

PlayerLogic

PlayerLogicBase

+ SelectedPiece : int
+ points : int

+ ChangeSelectedPiece(Selection : int)
+ addPoint()

SimpleGameMode

+ CheckOutcome()

GeneralGameMode

+ CheckOutcome()

GameBoard

+ rows : int
+ cols : int
+ EndOfGame : bool
+ GameBoardWin : Fl_Double_Window*
+ BoardButton : Fl_Toggle_Button*

+ initwin()
+ SetBoardDimensions(x : int, y : int) : bool
+ DrawButtons(player1data : PlayerLogic*, player2data : PlayerLogic*, gameData : GameLogic*)
+ show()
+ hide()

GameLogic

+ CurrentTurn : int
+ turn : int
+ GameMode : int
+ Last_Player_Scored : int
+ EndGame : bool
+ rows : int
+ cols : int
+ CPUpresent : bool
+ CPUplayernum : int
+ SpacesPlayed : vector< filledSpace >
+ FoundSequences : vector< tempFilledSpace >
+ CPUlastXplayed : int
+ CPUlastYplayed : int

+ RotatePlayerTurn()
+ setGameMode(x : int)
+ addMovetoList(x : int, y : int, Piece : int, Button_Used : Fl_Toggle_Button*)
+ GetWinner(p1score : int, p2score : int) : int
+ SequenceFinder(rows : int, cols : int, Player1Data : PlayerLogic*, Player2Data : PlayerLogic*) : int
+ CheckOutcome()
+ ~ GameLogic() «destructor»
+ CPUseek(win : Fl_Double_Window*)
+ FindButton(win : Fl_Double_Window*, x : int, y : int) : Fl_Toggle_Button*
+ CheckIfScore(rows : int, cols : int) : bool
+ printAllSequences()
+ HandleButtonPlayed(Player1Data : PlayerLogic*, Player2Data : PlayerLogic*, win : Fl_Double_Window*)