CSSE1001 Assignment 3
Revised Design Document

October 2008

Name: Nicholas Hughes
Student Number: 41785636
Project Title: PolyominOhs!

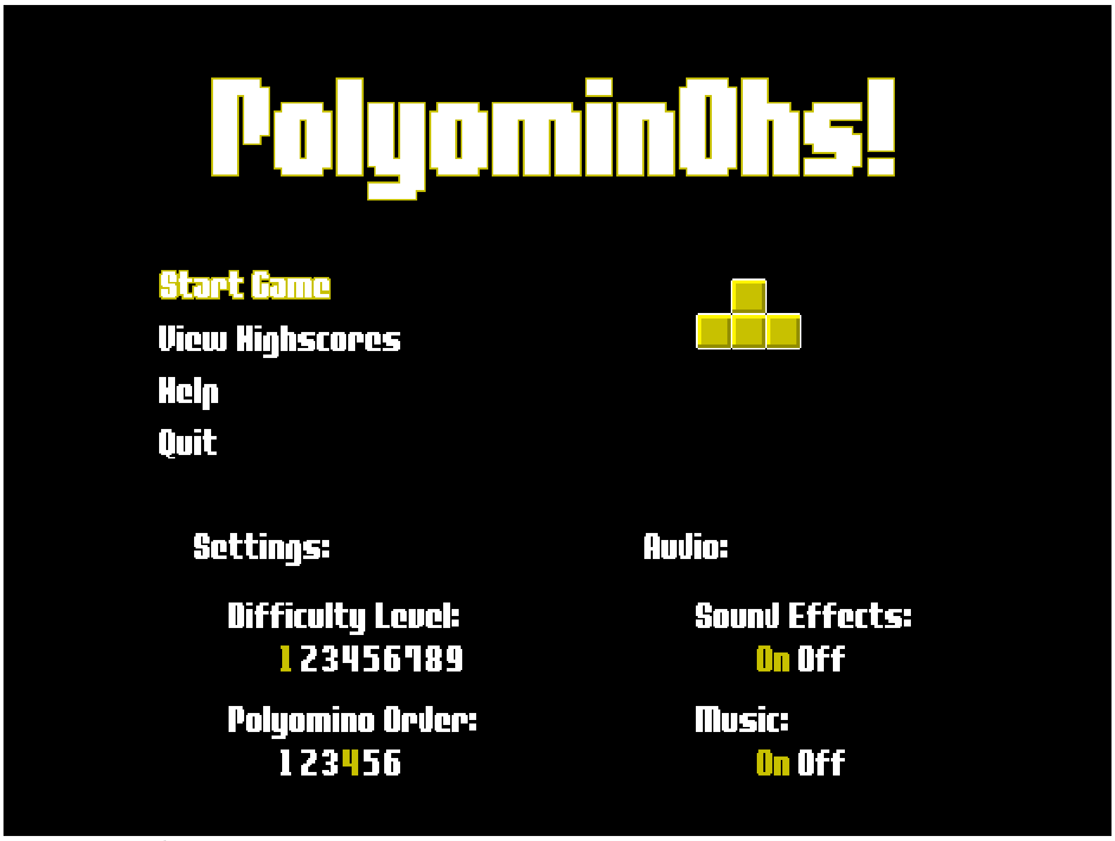
1. Description

The game of Tetris is based on 4-order polyominoes (tetronomines), which are shapes made up of 4 squares. The 7 blocks used in Tetris are all the one-sided polyominoes of order 4, where one-sided designates that polyominoes rotated or translated are identical (but ones reflected are not).

This document outlays a design for a game very similar to Tetris in game play, except that the order of polyominoes the user wishes the play with can be selected. In theory this design should be able to support any order polyominoes, however in the interests of game playability the game will be limited to orders 1 to 6. The game will procedurally generate the one-sided polyominoes of the order chosen by the user.

The menu screen allows the user to select polyomino order, difficulty level and audio option, as well as view high scores and help. Once the user starts a new game, it is nearly identical to Tetris. A count of how many lines of blocks the user has cleared is shown on screen, as is the player's score. The next n-omino coming for the user is also displayed next to the playing grid. The user uses the arrows keys to play the game.

2. User Interface



Settings:

Difficulty Level:
[✓ 1] 2 3 4 5 6 7 8 9

Polyomino Order:
1 2 3 [✓ 4] 5 6

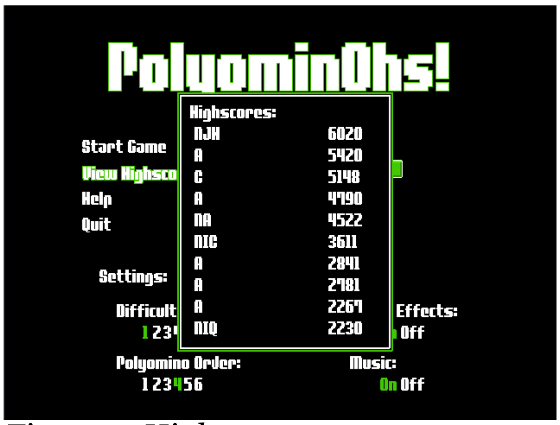
Audio:

Sound Effects:
[✓ On] Off

Music:
[✓ On] Off

Figure 1. The menu screen.

All user input is provided by the keyboard. On the menu screen shown in *figure 1* the up and down arrow keys allow selection of menu items. The left and right arrow keys allow the user to change the game settings and audio options, when those items are selected. Pressing enter on *View Highscores* displays a dialog box with the list of high scores (*figure 2*). Pressing enter on *Help* displays a dialog box explaining the game and listing the controls (*figure 3*). Pressing enter on *Start Game* starts a new game with the selected settings and options.



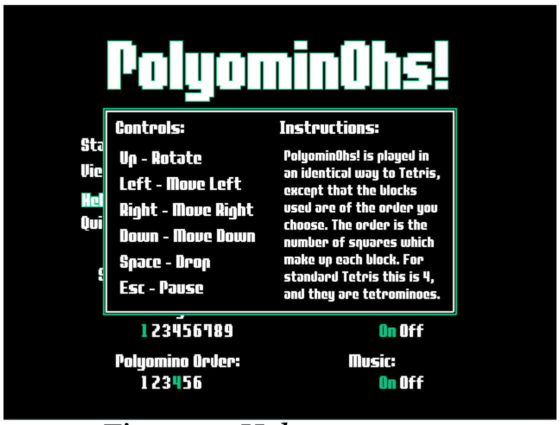
Highscores:

NJH	6020
A	5420
C	5148
A	4790
NA	4522
NIC	3611
A	2841
A	2781
A	2267
NIQ	2230

Polyomino Order:
1 2 3 [✓ 4] 5 6

Music:
[✓ On] Off

Figure 2. High scores screen.



Controls:

Up - Rotate
Left - Move Left
Right - Move Right
Down - Move Down
Space - Drop
Esc - Pause

Instructions:

PolyominOhs! is played in an identical way to Tetris, except that the blocks used are of the order you choose. The order is the number of squares which make up each block. For standard Tetris this is 4, and they are tetrominoes.

[✓ 1] 2 3 4 5 6 7 8 9 [✓ On] Off

Polyomino Order:
1 2 3 [✓ 4] 5 6

Music:
[✓ On] Off

Figure 3. Help screen.

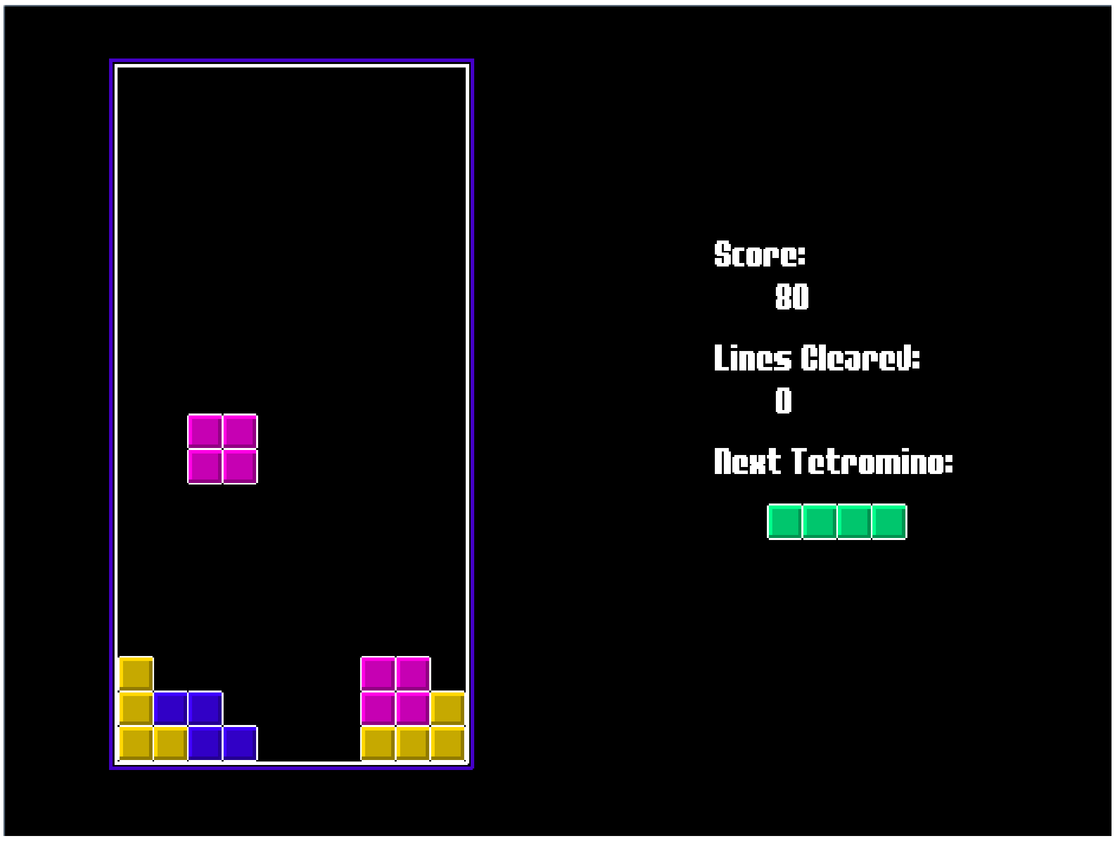


Figure 4. In game screen.

While playing the game (*figure 4*), the left and right arrow keys move the falling polyomino left and right. The up arrow key rotates the falling polyomino 90 degrees clockwise. The down arrow key moves the falling polyomino down faster. Pressing spacebar drops the falling polyomino instantly downwards, below its current position. Pressing escape displays the pause menu (*figure 5*), where one can press escape again to resume the game or the y key to quit back to the menu. If the playing grid fills up, the game over screen is displayed (*figure 6*).

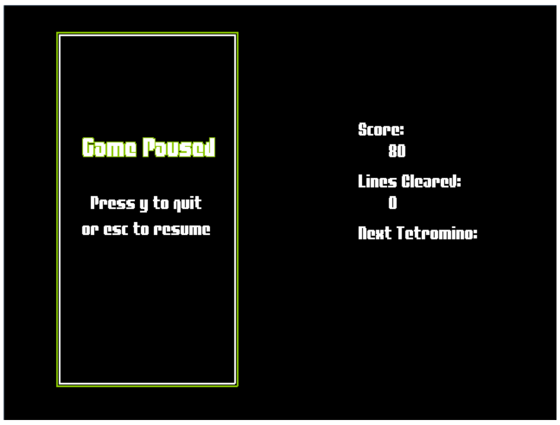


Figure 5. Pause screen.

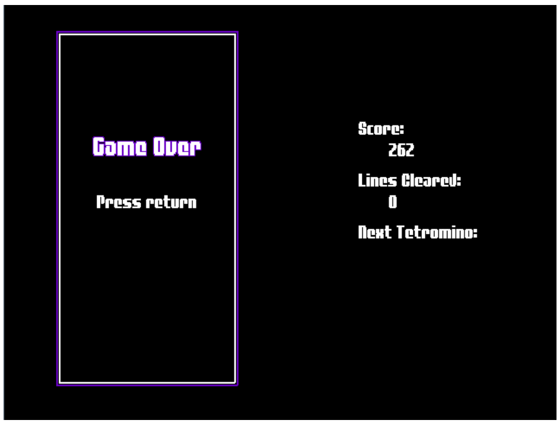


Figure 6. Game over screen.

If the score was sufficient to be placed on the high scores list, then following the game over screen, a dialog box allowing the user to enter their name will be displayed on the menu (*figure 7*). The arrow keys are used to select the three characters.

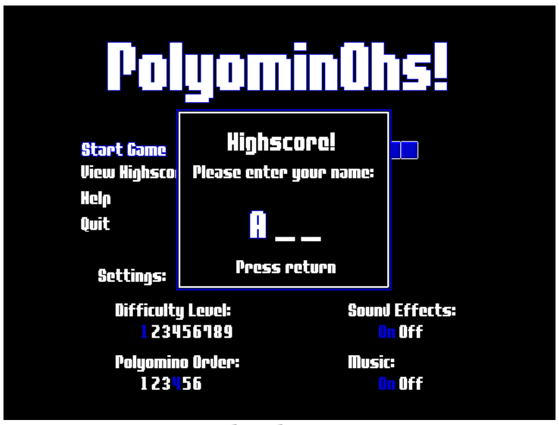


Figure 7. Enter high score name screen.

3. Design

The game is implemented in 12 module files, and the majority of the functionality is implemented through the main 9 classes, which are described below. A number of other ‘helper’ classes, such as 2D vectors and custom GUI widgets, are also used. Only the public methods of the classes are detailed here.

Class description template:

Class Name Class Description			
Constructor	Constructor Parameters (*type* name)		
Method Name	Method Parameters (*type* name)	Method Description	Return Variables (*type* name)

Main Classes:

Polyominohs The main application class which controls the application at the highest level.			
constructor			
run		The main entry point into the application.	
get_state		Returns the state of the application.	*int* state
change_state	*int* state	Changes the state of the application	
add_score	*string* name, *int* score	Adds the given score to the high scores list, and returns the index at which the score was added.	*int* index
get_highscores		Returns the high scores list.	*(string, int) pair list* highscores

Generator Generates polyominoes and colours.			
generate	*int* order	Returns a list of all the one-sided polyominoes of the given order.	*bool 2d list list* omiones
generate_colours	*int* number	Returns a list of the given number of RGB colours with evenly spaced hues.	*int triple list* colours

Field The playing field which contains a moving n-omino and a grid of blocks.			
constructor	*int* order, *int* width, *int* height		
get_size		Returns the size of the field: (width, height).	*int pair* size
get_complete_grid		Returns the grid with the moving n-omino baked into it.	*(bool, int triple) 2d list* grid
get_omino		Returns the currently moving n-omino.	*omino* omino
add_omino	*omino* omino	Attempts to add the given n-omino to the field and returns success.	*bool* added
move_omino	*int* direction	Attempts to move the moving n-omino in the given direction and returns success.	*bool* moved
rotate_omino		Attempts to rotate the moving n-omino and returns success.	*bool* rotated
check		Checks for full lines, clears them and returns the number of full lines cleared.	*int* full_lines

Omino A single polyomino. Handles its shape, colour, rotation and location in the field.			
constructor	*bool 2d list* shape*, int triple* colour*, int* rotation		
get_width		Returns the effective width of the omino.	*int* width
get_location		Returns the location of the omino.	*point* location
get_shape	*int* rotation	Returns the shape of the omino.	*bool 2d list* shape
get_rotation		Returns the rotation of the omino.	*int* rotation
get_colour		Returns the colour of the omino.	*int triple* colour
get_pivot		Returns the pivot point of the omino.	*point* pivot
get_offset	*int* rotation	Returns the offset for the given omino rotation.	*point* offset
move	*point* location	Moves the omino to the given location.	
rotate		Rotates the omino clockwise 90 degrees.	

Game The main game class which contains the game loop.			
constructor	*polyominohs* master, *view* view, *event_handler* event_handler, *sound* sound, *int* level, *int* order, *bool 2d list list* ominoes		
get_order		Returns the polyomino order of the game.	*int* order
get_score		Returns the current game score.	*int* score
get_lines_cleared		Returns the number of lines cleared in the game.	*int* lines_cleared
get_field		Returns the field object associated with the game.	*field* field
get_next_omino		Returns the n-omino which is coming up next.	*omino* omino
change_state	*int* state	Changes the state of the game.	
loop		The main game loop. Loops until the game is over, then returns the score.	*int* score

Menu The main menu clasws which contains the menu loop.			
constructor	*polyominohs* master, *view* view, *event_handler* event_handler, *sound* sound, *int* level, *int* order		
get_name_selected			*int* index
get_random_omino			
get_selection		Returns the currently selected menu item.	*int* selection
get_order		Returns the currently selected order option.	*int* order
get_level		Returns the currently selected difficulty level.	*int* level
get_sfx		Returns the currently selected sound effects option.	*string* sfx_selection
get_music		Returns the currently selected music option.	*string* music_selection
get_highscore_highlight		Returns the index of the high score to highlight on the high scores list.	*int* index
get_highscore_name			
change_state	*int* state	Changes the state of the menu.	
loop		The main menu loop. Loops until app quits or game starts, then returns selected options.	(*int* order, *int* level) *pair*

Event_Handler Provides an interface to pygame’s event system.			
constructor	*pygame* pygame		
clear_queue		Clears the event queue.	
get_events		Returns all the events which have occurred since last calling the function.	*pygame.event list* events

View Handles the visual component of the game.			
constructor	*pygame* pygame, *polyominohs* master		
change_state	*int* state, *game/menu* interface	Changes the state of the view by providing a new interface so the view knows what to display.	
start		Opens the window and starts the display.	
update		Updates the screen.	

Sound Handles the audio component of the game.			
constructor	*pygame* pygame		
get_music_on		Returns a string representing whether music is on or not.	*string* music_on
get_sound_effects_on		Returns a string representing whether sound effects are on or not.	*string* sfx_on
toggle_music		Toggles whether music is on or not.	
toggle_sound_effects		Toggles whether sound effects are on or not.	
play_next		Plays the next music track as well as setting an event to occur when the track stops.	
stop_music	*int* fadeout	Stops the music with an optional fade out time.	
play_sound_effect	*int* effect	Plays the specified sound effect.	

Other Classes:

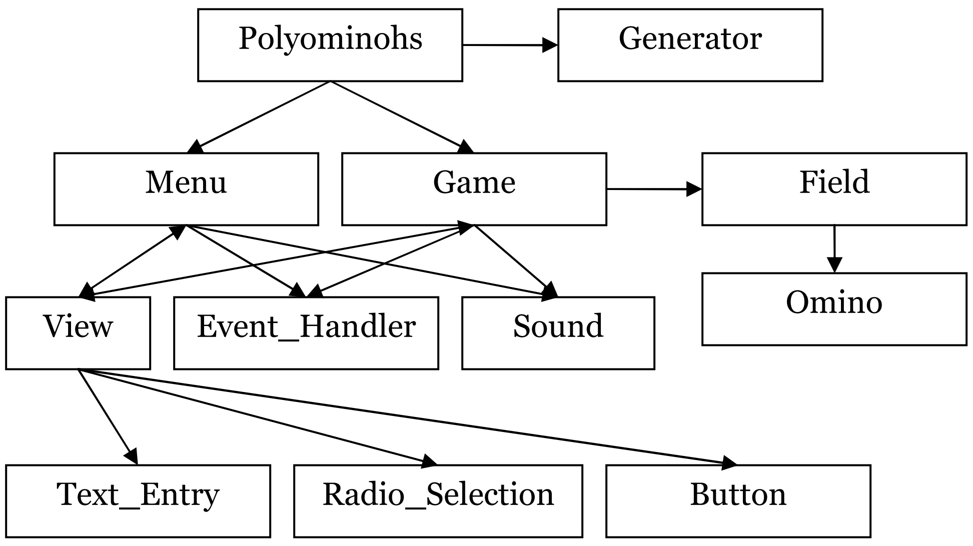
Point A two-dimensional position vector. All operators that make sense (e.g. addition, equality) are defined.		
constructor	*int* x, *int* y	Sets attributes x and y to the given values.

Text_Entry The visual component of a text entry field where one scrolls through the alphabet to select each character.			
constructor	*int* length, *char list* default, *int* size, *int pair* coords		
update	*char list* text	Updates the text to the given text.	
draw	*pygame.Surface* surface, *int* selected, *int triple* colour, *pygame* pygame	Draws the text entry field to the given surface with the given settings.	

Radio_Selection The visual component of a radio selection.			
constructor	*string list* items, *int* size, *int pair* coords		
draw	*pygame.Surface* surface, *int* selected, *int* mode, *int triple* colour, *pygame* pygame	Draws the radio selection to the given surface with the given settings.	

Button The visual component of a button.			
constructor	*string* text, *int* size, *int pair* coords		
draw	*pygame.Surface* surface, *int* mode, *pygame* pygame, *int triple* colour	Draws the button to the given surface with the given settings.	

Class Interaction:



As the class interaction diagram above shows, the Polyominohs class is the main class with which the whole application is controlled. The Menu and Game classes contain the main loops which run the individual aspects of the application. The View, Event_Handler and Sound classes provide interfaces to pygame's display, event and sound systems respectively, and these are used by the Game and Menu classes to run the application.

4. Support Modules

The copy, random, string, os and math modules, part of the standard Python library, are made use of. As well as this, the display, input and sound components of the game make extensive use of the pygame library. This library can be obtained from http://www.pygame.org/.