UNIVERSITY OF CANBERRA

ADVANCED GAME PROGRAMMING (9746)

ASSIGNMENT 2

ROBERT MAIR

U3157923

UNIT CONVENER: DR REZA RYAN

SEMESTER 1, 2020

Table of Contents

Table of Figures and Tables

Submission:

Students are to submit the complete and tested project by the due date. Students are expected to write their own code.

Submission requires your entire project in a ZIP file including:

Requirements:	Completed (Y/N) (notes)
Source code	YES – C#, using Visual Studio 2019
Console Application project	YES – Console
Technical design document (TDD),	YES
Screen Capture Video of your completed project	YES

PLEASE ACCESS CODE FROM GITHUB:

Link: https://github.com/RobertMair/u3157923_9746_Assessment2_2020

Assessment Criteria/Rubric:

1. Demonstrates a critical understanding of the key concepts covered in the unit lectures, tutorials, and workshops including the use of complex techniques such as Procedural Content Generation, Artificial Intelligence, Multiplayer Networking, and Shader programming.

   COMMENTS: Attempted to utilise as many of the courseware concepts and study material into the prototype game. The focus was on Artificial Intelligence using pathfinding (A*) and decision trees using 'weighted random'.

2. Provides evidence of using advanced programming techniques and methods such as appropriate data structures (e.g. List, Dictionary, and Hash table , Stacks, Queues, and Collection), LINQ Queries, Lambda expressions, delegate, inheritance, interface, abstract class, generic class, polymorphism, and modularization of code into specific classes and methods.

   COMMENTS: Attempts have been made to showcase as many of these as possible: Methods, classes, inheritance, variable/data types, user input, logical and mathematical calculations, collections (including 'object pooling'), loops and conditionals, branching etc. have.

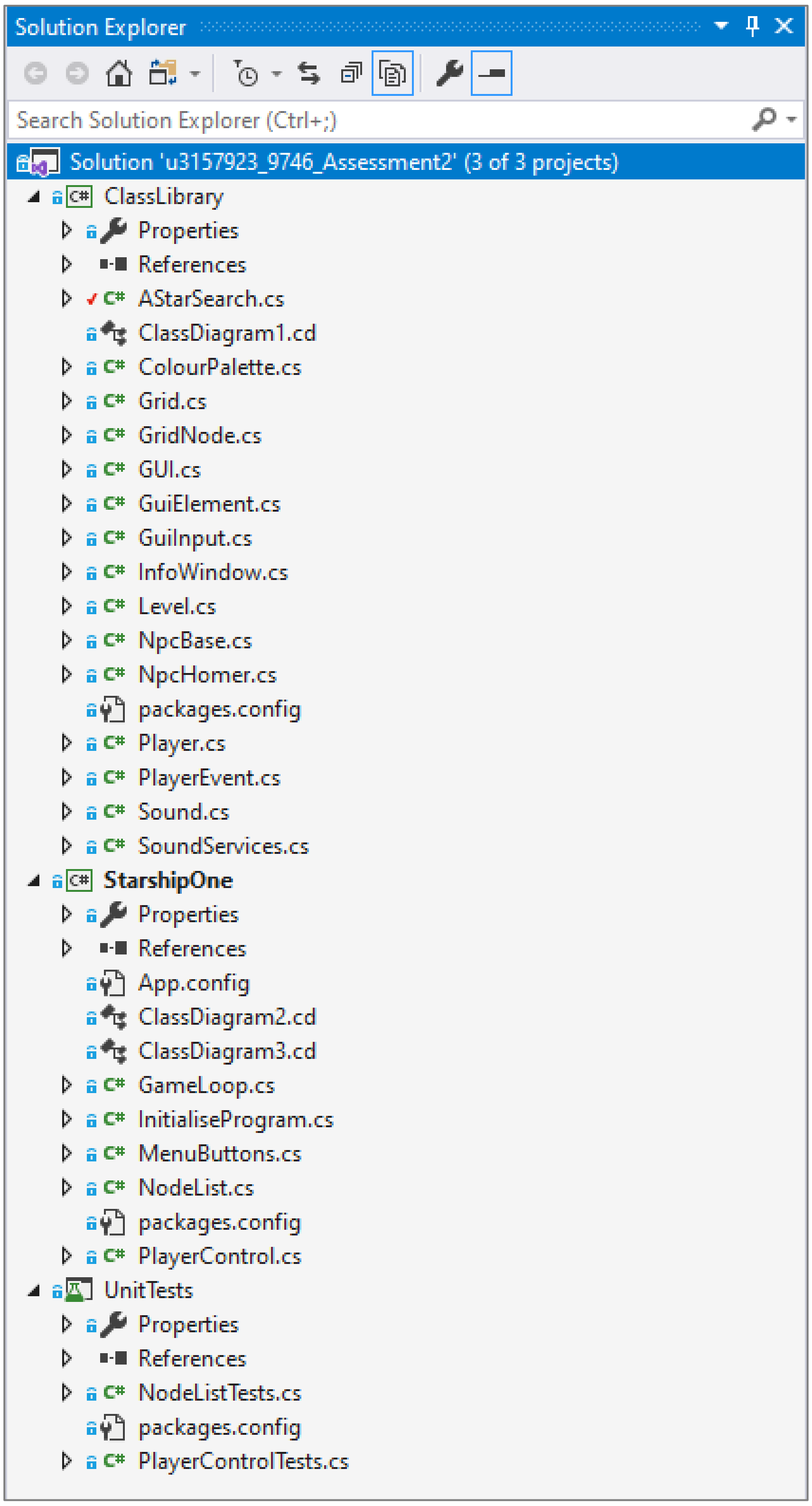
   Examples of advanced programming techniques used:

   - Data structures:

List:	NodeList.cs	Line 8
ENUM:	PlayerEvent.cs	Line 3
	PlayerControls.cs	Line 27
	GameLoop.cs	Line 107
Queues:	InfoWindow.cs	Line 9
Collection (objects):	GridNode.cs	Line 53

▪ Object instance clone:	GridNode.cs	Line 46
o LINQ Queries:	Level.cs	Line 224
o Lambda expressions:	AStarSearch.cs	Line 33
o Inheritance:	NpcBase.cs > NpcHomer.cs	
o Interface:	GridNode.cs	Line 6
o Abstract class:	NpcBase.cs	Line 3
o Polymorphism:	NpcBase.cs	Line 19
	NpcHomer.cs	Line 16

- Modularization of code into specific classes and methods:



```
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'u3157923_9746_Assessment2' (3 of 3 projects)
  ClassLibrary
    Properties
    References
    AStarSearch.cs
    ClassDiagram1.cd
    ColourPalette.cs
    Grid.cs
    GridNode.cs
    GUI.cs
    GuiElement.cs
    GuiInput.cs
    InfoWindow.cs
    Level.cs
    NpcBase.cs
    NpcHomer.cs
    packages.config
    Player.cs
    PlayerEvent.cs
    Sound.cs
    SoundServices.cs
  StarshipOne
    Properties
    References
    App.config
    ClassDiagram2.cd
    ClassDiagram3.cd
    GameLoop.cs
    InitialiseProgram.cs
    MenuButtons.cs
    NodeList.cs
    packages.config
    PlayerControl.cs
  UnitTests
    Properties
    References
    NodeListTests.cs
    packages.config
    PlayerControlTests.cs
```

3. Demonstrate the appropriate use of systematically design a game which ended up with producing a technical design document (TDD).

   COMMENTS: The TDD is not complete in all areas. A lot of time was spent on researching ‘roque like’ role playing games (RPGs) and the design components of the GUI and play levels etc. The TDD therefore attempts to highlight attempts/examples of each section rather than provide a comprehensive TDD given the enormity of that requirement given the prototype game size and complexity.

4. The project has been suitably tested and either perform as expected or, if bugs are present, they have been identified in a test plan.

   COMMENTS: A lot of 'play testing' has been completed during the development of Starship One. While most 'bugs' that have been identified have been fixed, further testing, particularly 'unit testing' is warranted. Currently the game has no 'build errors' and only a few 'run time' glitches (not so much to do with logic, but rather with GUI refresh synchronization with the game loop timer.

5. Demonstrates advanced quality and clarity of code, including commenting, formatting and consistency.

   COMMENTS: Code requires more 'refactoring' as a lot of the code was based on prior assignments and a 'game loop' was retrofitted when a design decision was made to change from a 'turn based' RPG format (NCPs only move each time the player moves) to a 'real time' RPG format (NPCs move independently of the player, this required implementation of 'timers' into the code and refactoring the code such that GUI updates are synchronized with the timer trigger correctly. Given more time, proper refactoring of the code would be completed including the use of the excellent ReSharper from JetBrains[1].

6. Demonstrates that technical design document has an adequate and consistent use of the scholarly practice and followed the submission instructions.

[1] 2018. *ReSharper: The Visual Studio Extension for .NET Developers by JetBrains.* [ONLINE] Available at: https://www.jetbrains.com/resharper/. [Accessed 11 November 2018].

Game Analysis

Prototype Game Concept – *Starship One*

Starship One is a 'roguelike' game set on the fictional spaceship 'R/V[2] Starship One', a prototype, lightly armed military interstellar transport designed to seek out and analyse new worlds for their suitability for terraforming and populating in the Andromeda Galaxy, 2.5 million light years from Earth[i].

Starship One has been funded at great cost by all the nations on earth to seek new planets to colonise to stem the effects of the growing population on Earths resources.

Starship One is designed to survive decades in space with a small crew and scientific, engineering and other speciality passengers required to locate new worlds for population. To support the human crew and passengers, the ship, comprising of a number of deck levels and rooms per deck to house living quarters, food production, engineering, engine rooms, sleeping quarters, air/water purification and recycling etc., has been equipped with a 'super computer' named Starguard to oversee ship systems and manage the large number of automatons (onboard robots to support the human crew).

During a routine inspection prior to commissioning[3], Starguard malfunctions and initiates Starship One's initial shakedown flight while also sending out scrambled instructions to the onboard automatons causing them to adopt a range of unexpected behaviours. This included scattering many of the ZnCl2 (Zinc-Chloride) cells that Starship One uses to power the many onboard systems.

Starguard, whose built-in redundancies initiated a reboot, then progressed beyond the 'shakedown' flight protocol, and has commenced the flight path to the Andromeda Galaxy. Realising that many ZnCl2 cells will need to be found so that Starship One has enough power for the journey, Starguard has managed to re-establish control of various automatons and has sent these out to locate the scattered ZnCl2 cells.

When Starship One launched, Riley ETO (Electro Technological Officer, a licensed member of the engine department of the ship as per Section A-III/6 of the STCW Code[ii]) was the only human onboard and was carrying out some routine tasks. Having checked an information portal regarding the ships systems, Riley learns what has occurred and realises he must seek out as many of the scattered ZnCl2 cells as possible in an attempt to re-establish control of Starship One and return the starship back to Earth for repairs.

Hence, the object of the game is to find and pick up as many collectables (including ZnCl2 cells) in each room of each deck. Progress is gained by collection enough ZnCl2 cells to access the next section of the deck, or the lift to the next deck. At the same time as finding collectibles, Riley must also avoid, disable, or reconfigure the various automatons. Eventually Riley will reach the main deck and CIC (Combat Information Centre) whereby the remaining collected ZnCl2 cells can be used to regain control of Starship One.

[2] Research Vessel.

[3] The act or ceremony of placing a ship in active service.

Game Design Summary[iii]

Game Identity / Mantra:

A game set in a research spaceship about an engineer trying to re-gain control of the ship from the onboard computer in order to return to earth.

Design Pillars:

Immersive. Continued sense of reward (achievement, progress). Logic puzzles combined with random elements (non-player character behaviours).

Genre/Story/Mechanics Summary:

Roque like RPG[iv] using basic tile-based ASCII graphics to allow the player to traverse the spaceship, solve puzzles, collect objects and disable non-playing characters (NPCs). The game will use a combination of Rogue like elements, such as animation/action (interaction with NPC's) mixed with problem solving (puzzles, such as opening doors via a combination of actions) and player rewards (objects used to progress to next level) to engage the player on a number of levels.

Interface:

Player input method:	Keyboard (Up, Down, Left, Right).
	Mouse to use 'menu' system.

Art Style:

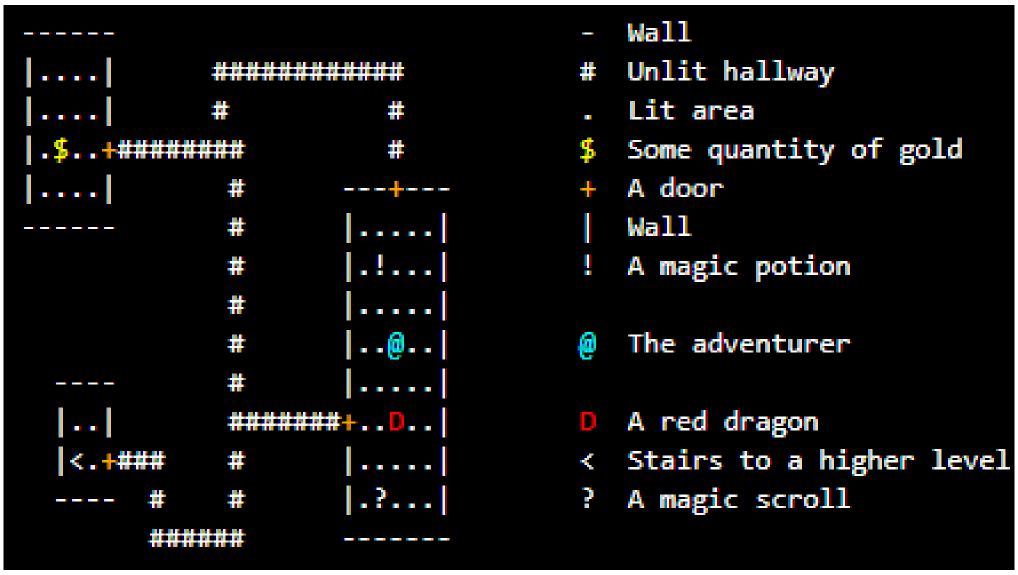
8-Bit, ASCII Style

Display is limited to specification of many 8-bit computers from the 1980's e.g. colours and character (rows/columns) resolution (although the game will be running in a 'console app' using C#).

Example 8-bit computers:

- Amstrad CPC[v]
- Commodore 64[vi]
- ZX Spectrum[vii]

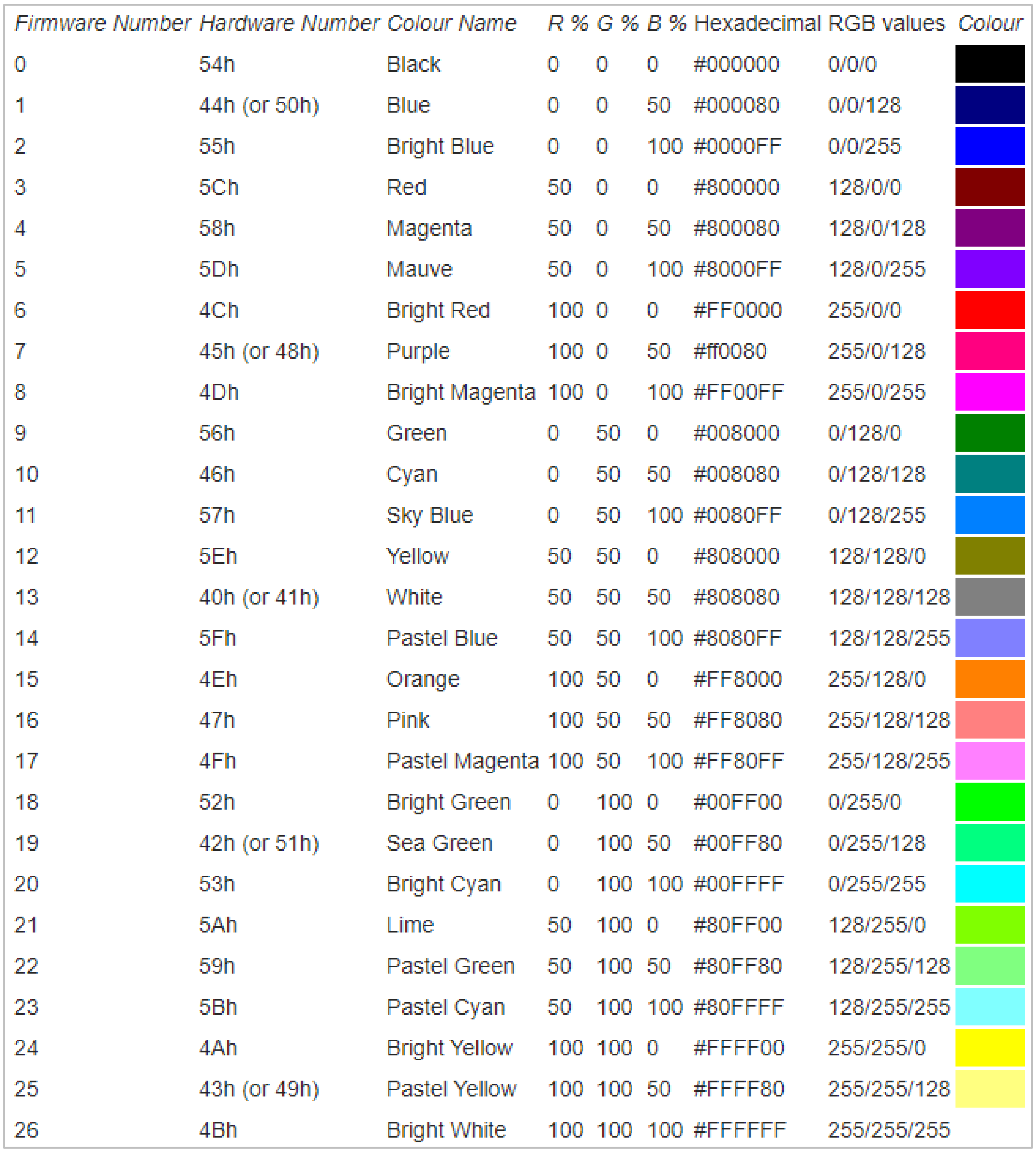
ASCII is a type of display used in Roguelike games. The typical ASCII display looks like the following[viii]:



```
------                              -  Wall
|....|      #############           #  Unlit hallway
|....|      #           #           .  Lit area
|.$..+#########         #           $  Some quantity of gold
|....|       #      ---+---         +  A door
------       #      |.....|         |  Wall
             #      |.!...|         !  A magic potion
             #      |.....|
             #      |..@..|         @  The adventurer
  ----       #      |.....|
  |..|       #######+..D..|         D  A red dragon
  |<.+###    #      |.....|         <  Stairs to a higher level
  ----  #    #      |.?...|         ?  A magic scroll
        ######      -------
```

Colour Schema

For this game, set console colours based on use the Amstrad CPC ‘Mode 1’ which allowed four colours from the following Amstrads colour palette of 27[ix]:



Firmware Number	*Hardware Number*	*Colour Name*	*R %*	*G %*	*B %*	Hexadecimal	RGB values	*Colour*
0	54h	Black	0	0	0	#000000	0/0/0	⬛
1	44h (or 50h)	Blue	0	0	50	#000080	0/0/128	🟦
2	55h	Bright Blue	0	0	100	#0000FF	0/0/255	🟦
3	5Ch	Red	50	0	0	#800000	128/0/0	🟥
4	58h	Magenta	50	0	50	#800080	128/0/128	🟪
5	5Dh	Mauve	50	0	100	#8000FF	128/0/255	🟪
6	4Ch	Bright Red	100	0	0	#FF0000	255/0/0	🟥
7	45h (or 48h)	Purple	100	0	50	#ff0080	255/0/128	🟥
8	4Dh	Bright Magenta	100	0	100	#FF00FF	255/0/255	🟪
9	56h	Green	0	50	0	#008000	0/128/0	🟩
10	46h	Cyan	0	50	50	#008080	0/128/128	🟦
11	57h	Sky Blue	0	50	100	#0080FF	0/128/255	🟦
12	5Eh	Yellow	50	50	0	#808000	128/128/0	🟨
13	40h (or 41h)	White	50	50	50	#808080	128/128/128	⬜
14	5Fh	Pastel Blue	50	50	100	#8080FF	128/128/255	🟦
15	4Eh	Orange	100	50	0	#FF8000	255/128/0	🟧
16	47h	Pink	100	50	50	#FF8080	255/128/128	🟥
17	4Fh	Pastel Magenta	100	50	100	#FF80FF	255/128/255	🟪
18	52h	Bright Green	0	100	0	#00FF00	0/255/0	🟩
19	42h (or 51h)	Sea Green	0	100	50	#00FF80	0/255/128	🟩
20	53h	Bright Cyan	0	100	100	#00FFFF	0/255/255	🟦
21	5Ah	Lime	50	100	0	#80FF00	128/255/0	🟩
22	59h	Pastel Green	50	100	50	#80FF80	128/255/128	🟩
23	5Bh	Pastel Cyan	50	100	100	#80FFFF	128/255/255	🟦
24	4Ah	Bright Yellow	100	100	0	#FFFF00	255/255/0	🟨
25	43h (or 49h)	Pastel Yellow	100	100	50	#FFFF80	255/255/128	🟨
26	4Bh	Bright White	100	100	100	#FFFFFF	255/255/255	⬜

Console Colours

In ConsoleColors class (part of the ClassLibrary), a list of foreground/background colours is crated per the following. When passing a .txt file or string to RenderTextToConsole or TextWrite methods of the GUI class, the ASCII code, if prefixed by the tilde ‘`’ character, will implement the Action e.g. a change of colour, or a Console.WriteLine etc.

See: https://www.dotnetperls.com/console-color

See: http://colorfulconsole.com - C# library.

See: https://stackoverflow.com/questions/7937256/custom-text-color-in-c-sharp-console-application to change foreground/background colours to custom colours.

ASCII Code	ASCII Chr	Case #	Action	
33	!	0	Console.ForegroundColor =	⬛
34	"	1	Console.ForegroundColor =	🟦
35	#	2	Console.ForegroundColor =	🟦
36	$	3	Console.ForegroundColor =	🟥
37	%	4	Console.ForegroundColor =	🟪
38	&	5	Console.ForegroundColor =	🟪
39	'	6	Console.ForegroundColor =	🟥
40	(	7	Console.ForegroundColor =	🟥
41	)	8	Console.ForegroundColor =	🟪
42	*	9	Console.ForegroundColor =	🟩
43	+	10	Console.ForegroundColor =	🟦
44	,	11	Console.ForegroundColor =	🟦
45	-	12	Console.ForegroundColor =	🟫
46	.	13	Console.ForegroundColor =	⬛
47	/	14	Console.ForegroundColor =	🟦
48	0	15	Console.ForegroundColor =	🟧
49	1	16	Console.ForegroundColor =	🟥
50	2	17	Console.ForegroundColor =	🟪
51	3	18	Console.ForegroundColor =	🟩
52	4	19	Console.ForegroundColor =	🟩
53	5	20	Console.ForegroundColor =	🟦
54	6	21	Console.ForegroundColor =	🟩
55	7	22	Console.ForegroundColor =	🟩
56	8	23	Console.ForegroundColor =	🟦
57	9	24	Console.ForegroundColor =	🟨
58	:	25	Console.ForegroundColor =	🟨
59	;	26	Console.ForegroundColor =	⬜

ASCII Code	ASCII Chr	Case #	Action
60	<	27	Console.BackgroundColor = ⬛
61	=	28	Console.BackgroundColor = 🟦
62	>	29	Console.BackgroundColor = 🟦
63	?	30	Console.BackgroundColor = 🟥
64	@	31	Console.BackgroundColor = 🟪
65	A	32	Console.BackgroundColor = 🟪
66	B	33	Console.BackgroundColor = 🟥
67	C	34	Console.BackgroundColor = 🟥
68	D	35	Console.BackgroundColor = 🟪
69	E	36	Console.BackgroundColor = 🟩
70	F	37	Console.BackgroundColor = 🟦
71	G	38	Console.BackgroundColor = 🟦
72	H	39	Console.BackgroundColor = 🟫
73	I	40	Console.BackgroundColor = ▨
74	J	41	Console.BackgroundColor = 🟦
75	K	42	Console.BackgroundColor = 🟧
76	L	43	Console.BackgroundColor = 🟥
77	M	44	Console.BackgroundColor = 🟪
78	N	45	Console.BackgroundColor = 🟩
79	O	46	Console.BackgroundColor = 🟩
80	P	47	Console.BackgroundColor = 🟦
81	Q	48	Console.BackgroundColor = 🟩
82	R	49	Console.BackgroundColor = 🟩
83	S	50	Console.BackgroundColor = 🟦
84	T	51	Console.BackgroundColor = 🟨
85	U	52	Console.BackgroundColor = 🟨
86	V	53	Console.BackgroundColor =
87	W	54	Console.WriteLine();
88	X	55	Console.Clear();

Using initial colour schema:

	#FF8000 255/128/0	#FFFF80 255/255/128	#008080 0/128/128	#000000 0/0/0
FG:	15	25	10	0
BG:	42	52	37	27

Using final colour schema:

	#FF8000 255/0/0	#FFFF80 255/255/000	#008080 0/0/255	#000000 0/0/0
FG:	12	14	9	0
BG:	12	14	9	0

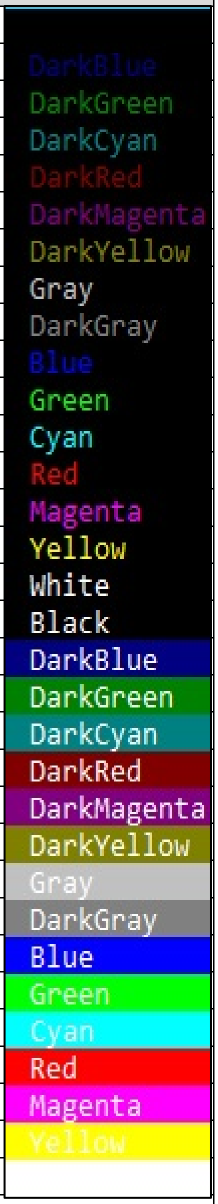
**** Colour Schema Update ****

HOWEVER, the custom colour palette was abandoned when it was discovered that it does not operate on Windows when running under emulation on a Macintosh computer, hence the standard Console colour schema, as outlined below, has been used.

See: https://www.dotnetperls.com/console-color

See: http://colorfulconsole.com - C# library.

ASCII Code	ASCII Chr	Case #		Action
33	!	0	0	**ForegroundColor = Black**
34	"	1	1	ForegroundColor = DarkBlue
35	#	2	2	**ForegroundColor = DarkGreen**
36	$	3	3	ForegroundColor = DarkCyan
37	%	4	4	**ForegroundColor = DarkRed**
38	&	5	5	ForegroundColor = DarkMagenta
39	'	6	6	ForegroundColor = DarkYellow
40	(	7	7	ForegroundColor = Gray
41	)	8	8	ForegroundColor = DarkGray
42	*	9	9	ForegroundColor = Blue
43	+	10	10	ForegroundColor = Green
44	,	11	11	ForegroundColor = Cyan
45	-	12	12	ForegroundColor = Red
46	.	13	13	ForegroundColor = Magenta
47	/	14	14	**ForegroundColor = Yellow**
48	0	15	15	ForegroundColor = White
49	1	16	0	**BackgroundColor = Black**
50	2	17	1	BackgroundColor = DarkBlue
51	3	18	2	**BackgroundColor = DarkGreen**
52	4	19	3	BackgroundColor = DarkCyan
53	5	20	4	**BackgroundColor = DarkRed**
54	6	21	5	BackgroundColor = DarkMagenta
55	7	22	6	BackgroundColor = DarkYellow
56	8	23	7	BackgroundColor = Gray
57	9	24	8	BackgroundColor = DarkGray
58	:	25	9	BackgroundColor = Blue
59	;	26	10	BackgroundColor = Green
60	<	27	11	BackgroundColor = Cyan
61	=	28	12	BackgroundColor = Red
62	>	29	13	BackgroundColor = Magenta
63	?	30	14	**BackgroundColor = Yellow**
64	@	31	15	BackgroundColor = White
65	A	32		Console.WriteLine();
66	B	33		Console.Clear();



ASCII Character Set

The extended ASCII character set shall be used in Starship One.

See https://theasciicode.com.ar/extended-ascii-code/black-square-ascii-code-254.html

Player Character

Type	ASCII	Details
Riley ETO	@ @	Player character, movement based on player input: up, down, left, right, activate etc. Key CharCode Up 38 w 87 Down 40 s 83 Left 37 a 65 Right 39 d 68

Automatons (AI's)

Type	ASCII	Details
Guard	G G	Predefined paths, typically to protect collectibles (until a Drone can collect it), doors, maintenance and communication terminals etc.
Berserker	B B	Movement is random, may run into Riley or a collectible but don't actively look for them.
Seeker	S S	Look for the Riley using search algorithms, difficult to avoid, not interested in collectibles.
Homer	H H	Will target Riley as soon as entering a room based on his location. They have advanced SAR (Search And Rescue) technology, however they are not benevolent and will target Riley until they are disabled.
Drone	D D	Look for collectibles, drones can work together when there are more than one in the room. They will seek the exit once all collectibles are found. Riley can intercept a drone and retrieve collectibles from it.

Collectibles

Type	ASCII	Details
ZnCl2 cell	z z	Collect ZnCl2 cells to unlock the section/level.
Energy cell	e e	Increase energy of the player space suite (energy will reduce when disabling automatons or encountering dangerous terrain).
Oxygen cell	o o	Increase oxygen tank level of the player space suite (oxygen will reduce when the player moves).
Tool	t t	Tools that can be used to disable automatons / repair damage.
Space case	c c	A sturdy storage container that can survive the vacuum of space. Don't know what is in a space case, can be a collectible, or booby trap

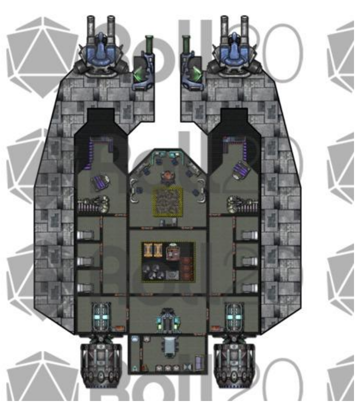
Terrain

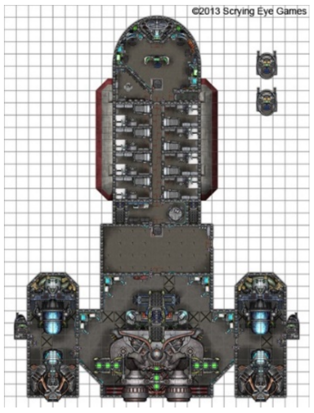
Type	ASCII	Details
Space	- ⬛	Space (outside of the spaceship).
Star		A star in space (outside of the spaceship).
Deck floor	. 🟦	Walkable.
Wall	# 🟧	Walls within the spaceship.
Door	+ +	Between rooms/corridors etc.
Locked door	* *	Between rooms/corridors etc. Must be unlocked to pass through.
Info Terminal	? ?	Player can unlock doors, gain information etc.
Acid	, ,	Walkable, but will reduce player space suite energy.
Ice	~ ~	Unless wearing 'ice' spikes, will cause player to slide in direction they stepped onto the ice.

Type	ASCII	Details
Force field	_ | _ |	Can block corridors, reduce player space suite energy to walk through (can they be deactivated?
One way (up)	^ ^	Player can move through this in one direction – up.
One way (down)	v v	Player can move through this in one direction – down.
One way (left)	< >	Player can move through this in one direction – left.
One way (right)	> <	Player can move through this in one direction – right.
Airlock (to next section)	A A	Airlock, used to seal each section of a level, operates only once enough ZnCl2 cells are found in the current section.
Lift (to next level)	L L	Lift to next level, operates only once enough ZnCl2 cells are found on the current level.

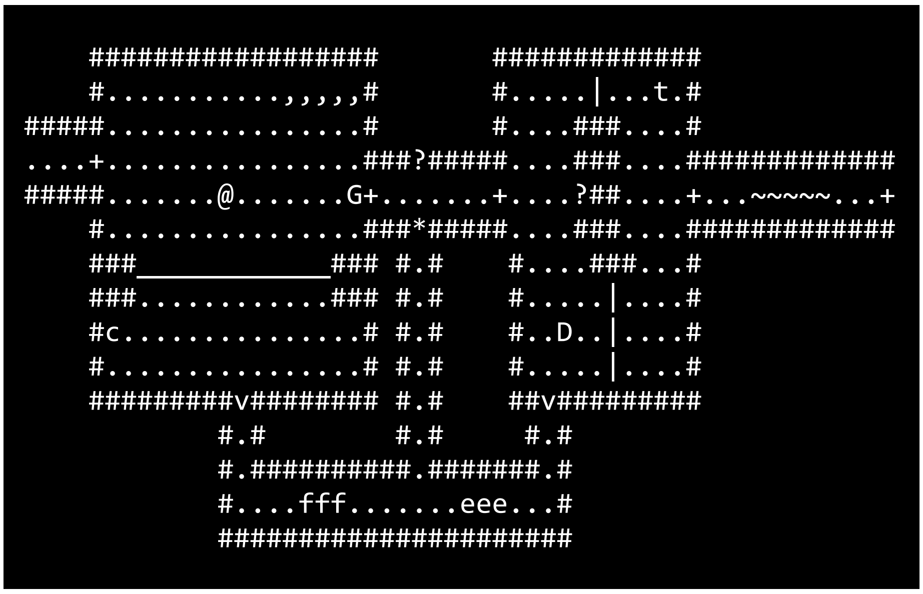
Example Spaceship Deck Designs (illustrations)[x]

The spaceship desk design is influenced by science fiction illustrations such as the following.





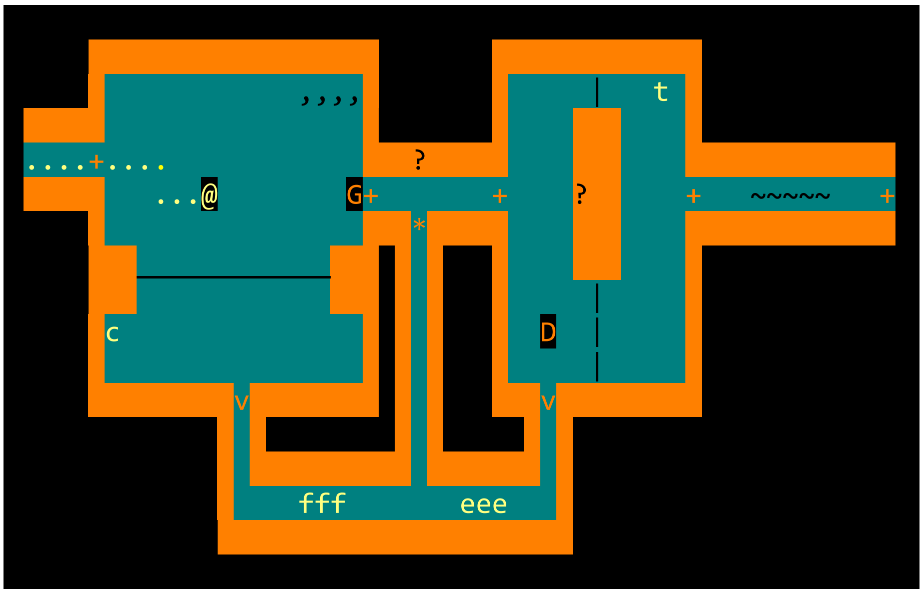
Example Deck Design (ASCII) – Early B&W Concept



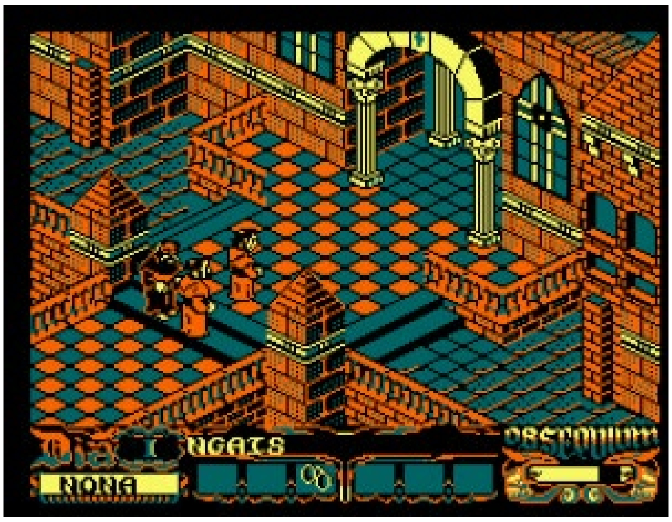
```
    ##################           #############
    #...........,,,,,#           #.....|...t.#
#####................#           #....###....#
....+................###?#####....###....#############
#####.......@.......G+.......+....?##....+...~~~~~...+
    #................###*#####....###....#############
    ###____________### #.#    #....###...#
    ###............### #.#    #.....|....#
    #c...............# #.#    #..D..|....#
    #................# #.#    #.....|....#
    #########v######## #.#    ##v#########
            #.#        #.#     #.#
            #.##########.#######.#
            #....fff.......eee...#
            ######################
```

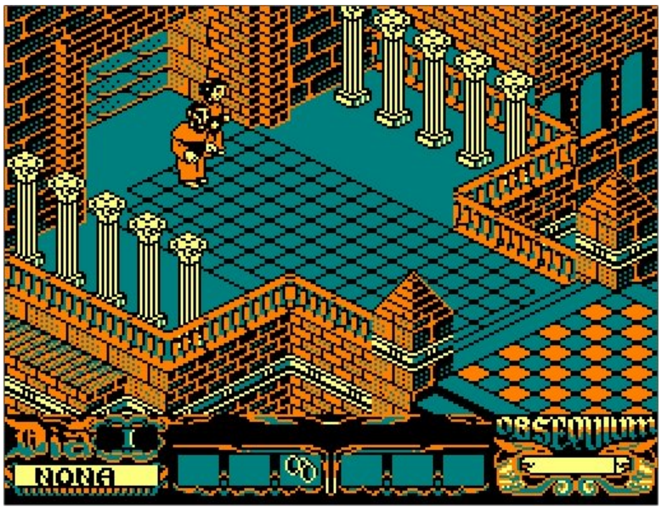
*Example Deck Design (ASCII) – Early Colour Concept**

`#FF8000 255/128/0` `#FFFF80 255/255/128` `#008080 0/128/128` `#000000 0/0/0`



*Colour palette based on the Amstrad CPC game LA ABADIA DEL CRIMEN[xi]





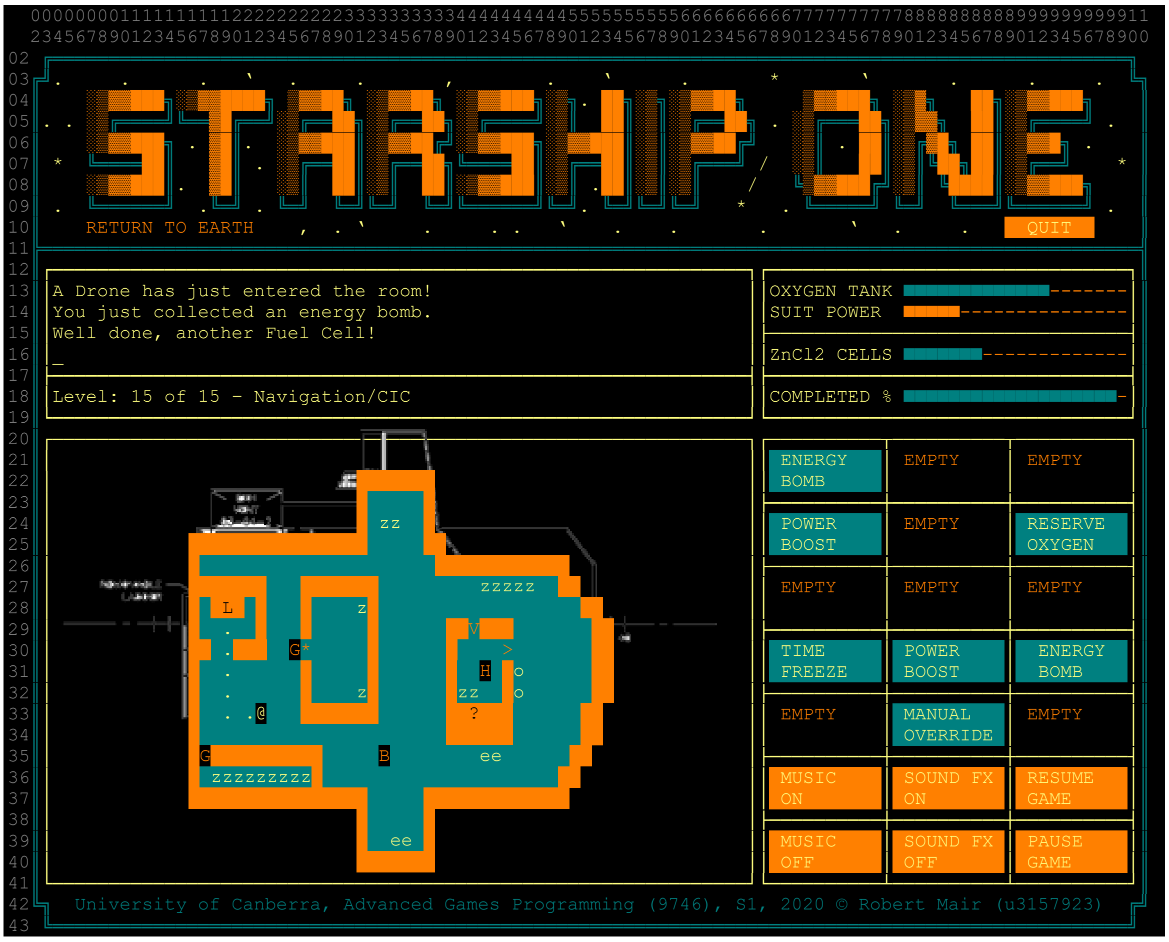
Level Design (using early colour concept):

User Interface

User interface design showcasing Level 04, the final level Riley must complete to finish the game.

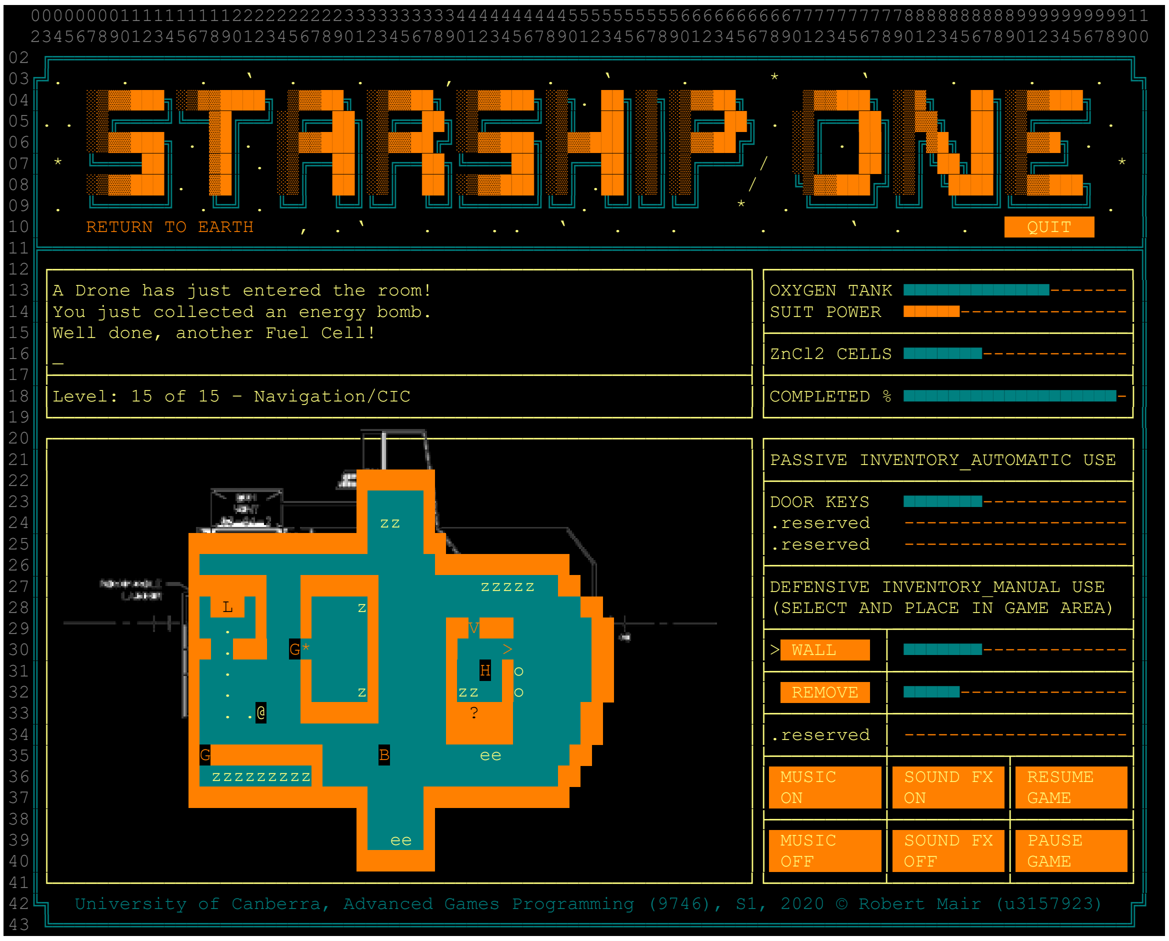
Design elements:

Version 1 design



```
RETURN TO EARTH                                                          QUIT

A Drone has just entered the room!              OXYGEN TANK
You just collected an energy bomb.              SUIT POWER
Well done, another Fuel Cell!
_                                               ZnCl2 CELLS

Level: 15 of 15 – Navigation/CIC                COMPLETED %

                                                ENERGY      EMPTY       EMPTY
                                                BOMB
                                                POWER       EMPTY       RESERVE
                                                BOOST                   OXYGEN
                                                EMPTY       EMPTY       EMPTY

                                                TIME        POWER       ENERGY
                                                FREEZE      BOOST       BOMB
                                                EMPTY       MANUAL      EMPTY
                                                            OVERRIDE
                                                MUSIC       SOUND FX    RESUME
                                                ON          ON          GAME
                                                MUSIC       SOUND FX    PAUSE
                                                OFF         OFF         GAME

University of Canberra, Advanced Games Programming (9746), S1, 2020 © Robert Mair (u3157923)
```

Version 2 design (final)



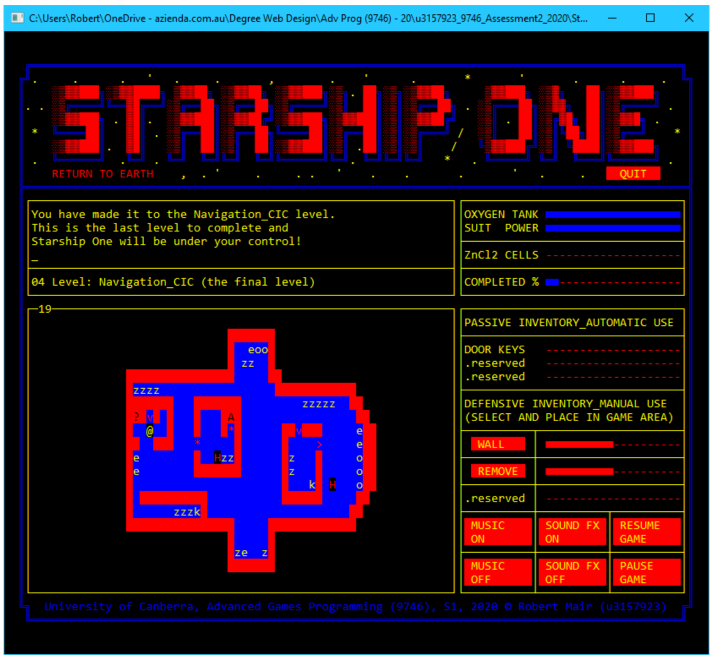
```
RETURN TO EARTH                                                QUIT

A Drone has just entered the room!              OXYGEN TANK
You just collected an energy bomb.              SUIT POWER
Well done, another Fuel Cell!
                                                ZnCl2 CELLS
_
Level: 15 of 15 - Navigation/CIC                COMPLETED %

                                                PASSIVE INVENTORY_AUTOMATIC USE

                                                DOOR KEYS
                                                .reserved
                                                .reserved

                                                DEFENSIVE INVENTORY_MANUAL USE
                                                (SELECT AND PLACE IN GAME AREA)

                                                > WALL

                                                  REMOVE

                                                .reserved

                                                MUSIC     SOUND FX   RESUME
                                                ON        ON         GAME

                                                MUSIC     SOUND FX   PAUSE
                                                OFF       OFF        GAME

University of Canberra, Advanced Games Programming (9746), S1, 2020 © Robert Mair (u3157923)
```

Using initial colour schema:

#FF8000 255/128/0	#FFFF80 255/255/128	#008080 0/128/128	#000000 0/0/0

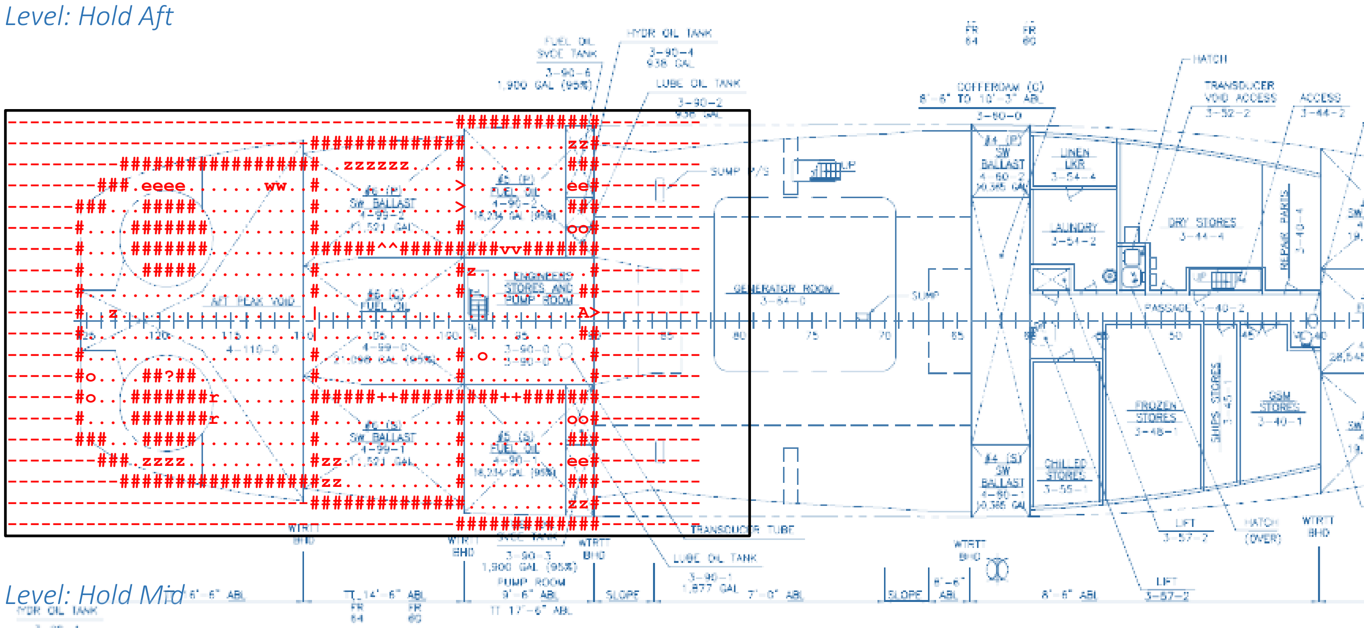
Using final colour schema:

#FF8000 255/0/0	#FFFF80 255/255/000	#008080 0/0/255	#000000 0/0/0

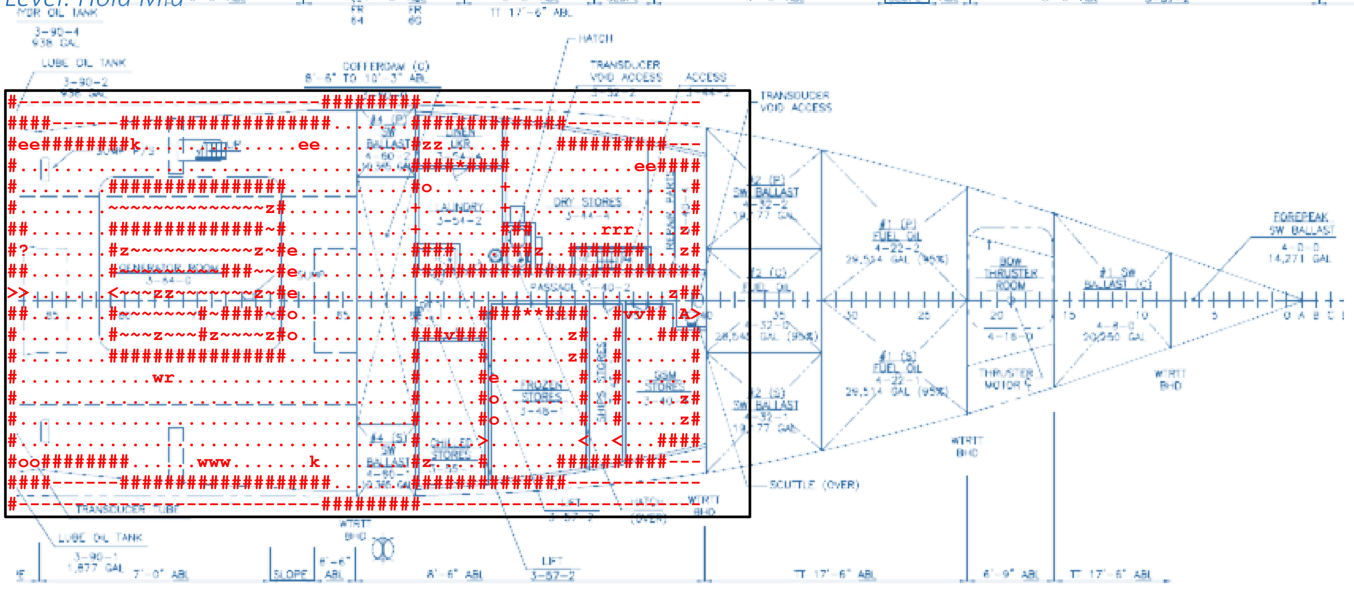


```
C:\Users\Robert\OneDrive - azienda.com.au\Degree Web Design\Adv Prog (9746) - 20\u3157923_9746_Assessment2_2020\St...

RETURN TO EARTH                                                QUIT

You have made it to the Navigation_CIC level.   OXYGEN TANK
This is the last level to complete and          SUIT  POWER
Starship One will be under your control!
                                                ZnCl2 CELLS
_
04 Level: Navigation_CIC (the final level)      COMPLETED %

19
                                                PASSIVE INVENTORY_AUTOMATIC USE

                                                DOOR KEYS
                                                .reserved
                                                .reserved

                                                DEFENSIVE INVENTORY_MANUAL USE
                                                (SELECT AND PLACE IN GAME AREA)

                                                WALL

                                                REMOVE

                                                .reserved

                                                MUSIC     SOUND FX   RESUME
                                                ON        ON         GAME

                                                MUSIC     SOUND FX   PAUSE
                                                OFF       OFF        GAME

University of Canberra, Advanced Games Programming (9746), S1, 2020 © Robert Mair (u3157923)
```

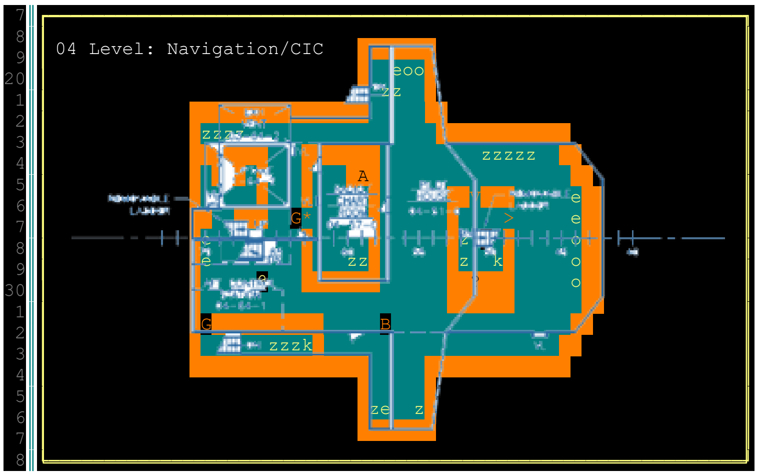
00 Level: Hold Aft

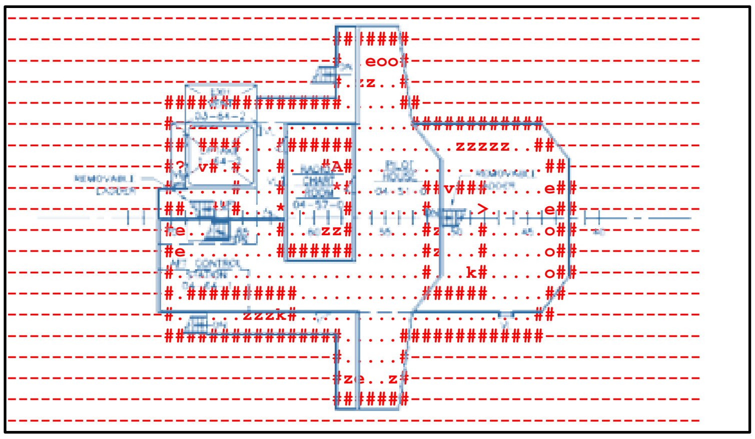


00 Level: Hold Mid



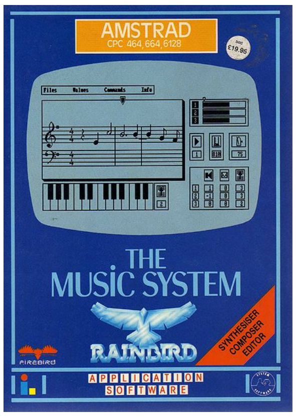
04 Level: Navigation_CIC (the final level)

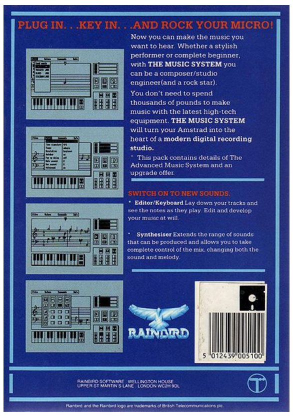


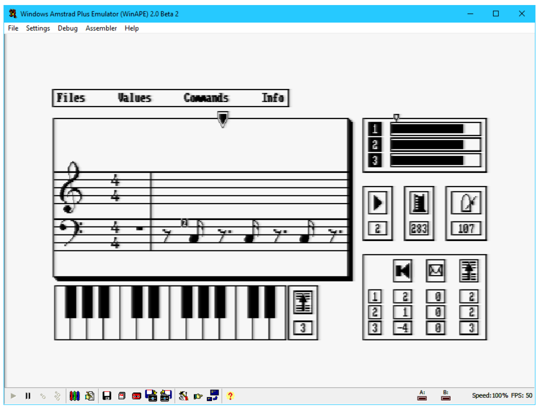


Music/Sound:

Use 8-bit sound effects and music, these can be created using such software as The Music System[xii] software for the Amstrad CPC range of computers.







However, due to time constraints, rather then compose music and create sound effects the prototype games uses:

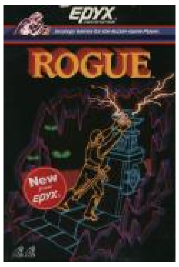
- Music by Eric Skiff (https://ericskiff.com/music/)[xiii]
- Sound Effects from Open Game Art (https://opengameart.org/content/512-sound-effects-8-bit-style)[xiv]

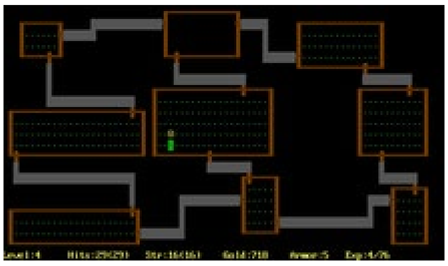
Influential Games

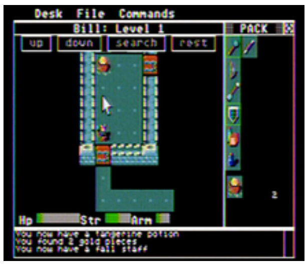
Rogue[xv]

Platform:	Amiga, Amstrad CPC, Atari 8-bit, Atari ST, Commodore 64, CP/M, DOS, Macintosh, TOPS-20, TRS-80 CoCo, Unix, ZX Spectrum
Year:	1980

Rogue (also known as Rogue: Exploring the Dungeons of Doom) is a dungeon crawling video game by Michael Toy and Glenn Wichman and later contributions by Ken Arnold. Rogue was originally developed around 1980 for Unix-based mainframe systems as a freely-distributed executable (public domain software). It was later included in the official Berkeley Software Distribution 4.2 operating system (4.2BSD). Commercial ports of the game for a range of personal computers have been made.





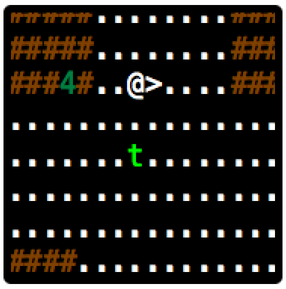


@ngband[xvi]

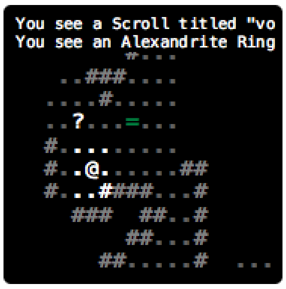
Platform: Windows, macOS (10.9+)

Year: 1980

Angband started life as a game called Moria in the 1980s in a programming language called VMS Pascal, later made the transition to C, as UMoria, and eventually become Angband some twenty years ago in 1994. To be honest, it's amazing that it's still going, and part of that is because there is an active and vibrant community of players and an always-changing set of active variants, or modified versions, many of which play very differently to Angband.

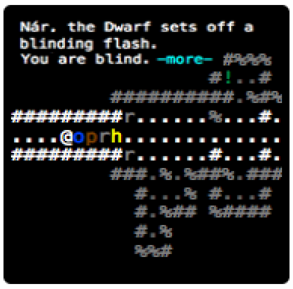


You (the @) begin in the small trading town above Angband, ready to descend into its depths (using the down staircase, '>').



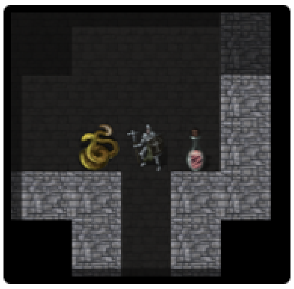
```
You see a Scroll titled "vo
You see an Alexandrite Ring
```

You explore randomly-generated caverns, and find treasure: = is a ring, ? is a scroll.



```
Nár, the Dwarf sets off a
blinding flash.
You are blind. -more-
```

You fight monsters. o is an orc, p a human, and r a rat. If you die in battle, you have to start the game again!



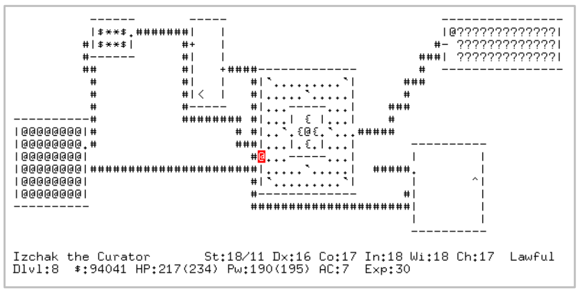
Graphics are also available if ASCII isn't your style.

Nethack[xvii]

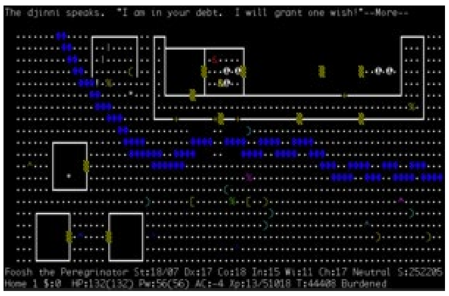
Platform: Various

Year: 1987

NetHack is a single player dungeon exploration game that runs on a wide variety of computer systems, with a variety of graphical and text interfaces all using the same game engine. Unlike many other D&D-inspired games, the emphasis in NetHack is discovering details of the dungeon and not killing everything - in fact, killing everything is a good way to die quickly. Each game presents a different landscape - the random number generator provides an essentially unlimited number of variations of the dungeon and its denizens to be discovered by the player in one of a number of characters: you can pick your race, your role, and your gender.



```
Izchak the Curator       St:18/11 Dx:16 Co:17 In:18 Wi:18 Ch:17  Lawful
Dlvl:8  $:94041 HP:217(234) Pw:190(195) AC:7  Exp:30
```



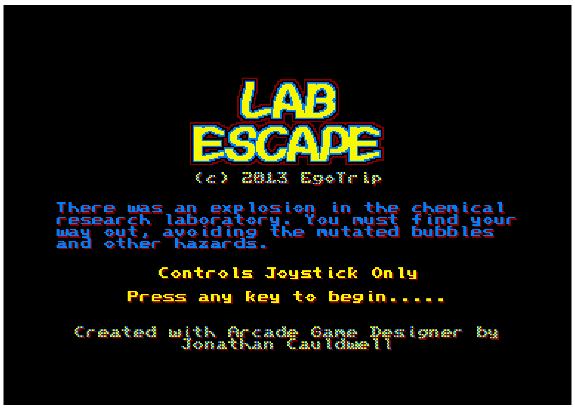
```
The djinni speaks.  "I am in your debt.  I will grant one wish!"--More--
Foosh the Peregrinator St:18/07 Dx:17 Co:18 In:15 Wi:11 Ch:17 Neutral S:252205
Home 1 $:0  HP:132(132) Pw:56(56) AC:-4 Xp:13/51018 T:44408 Burdened
```

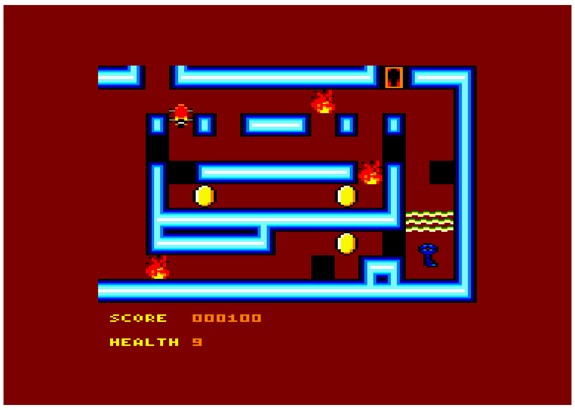
Lab Escape[xviii]

Platform: Amstrad CPC

Year: 2013

A mini adventure. Basically, you need to escape from the lab which exploded, and there are various hazards like fires, acid pools and mutated bubbles, collecting the money that got scattered around the lab.


LAB ESCAPE
(c) 2013 EgoTrip
There was an explosion in the chemical research laboratory. You must find your way out, avoiding the mutated bubbles and other hazards.
Controls Joystick Only
Press any key to begin.....
Created with Arcade Game Designer by Jonathan Cauldwell

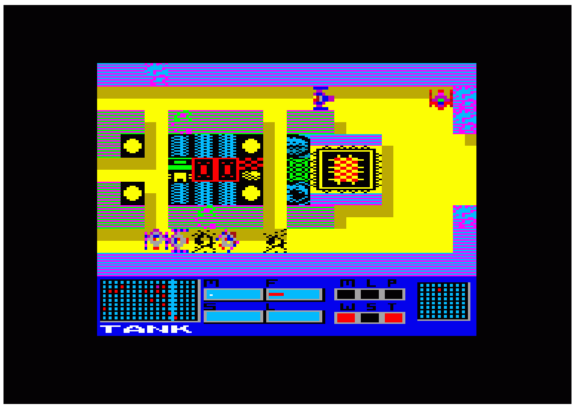


Panzadrome[xix]

Platform: Amstrad CPC

Year: 1985

The Panzadrome is a heavily fortified island guarded by robot tanks, and your aim is to destroy it completely. You begin with a poorly equipped tank, but there are factories on the island where you can upgrade it. However, you will have to get past armies of robot tanks and turrets and avoid any mines in your path in order to reach a factory – but achieving this is practically impossible. Moreover, all this destruction leaves craters in your way which your tank cannot drive over, and you could easily find your path blocked, unless you have some Polycrete to fill in the crater – but to obtain it, you must find the correct factory first! The graphics and sound effects are very poor and the game is far too difficult.

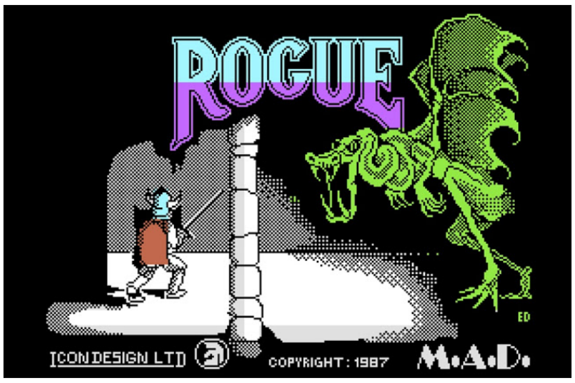


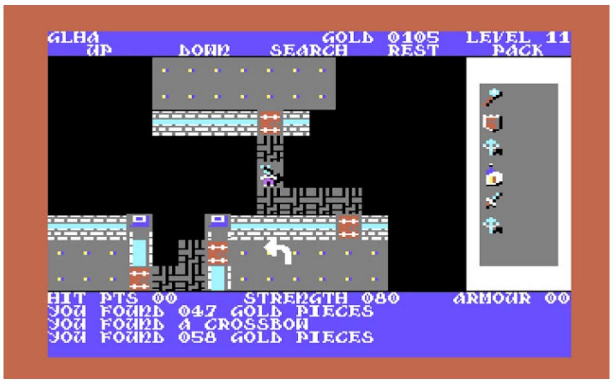
Rogue[xx]

Platform: Commodore 64

Year: 1985

A turn-based dungeon crawling roguelike, you are an adventurer that must explore the Dungeons of Doom, to retrieve the precious Amulet of Yendor! Designed using ASCII characters that represent locations, items and the main character, you must fight your way through many different dungeons to reach your end goal. Along the way you'll fight over 26 different monsters and come across levels that get much harder and more complex as you progress through the game. You can find great weapons of power, armour and even gain experience to level up and fight these horrific enemies that also become more difficult!





YOU FOUND 047 GOLD PIECES
YOU FOUND A CROSSBOW
YOU FOUND 058 GOLD PIECES

Ranarama[xxi]

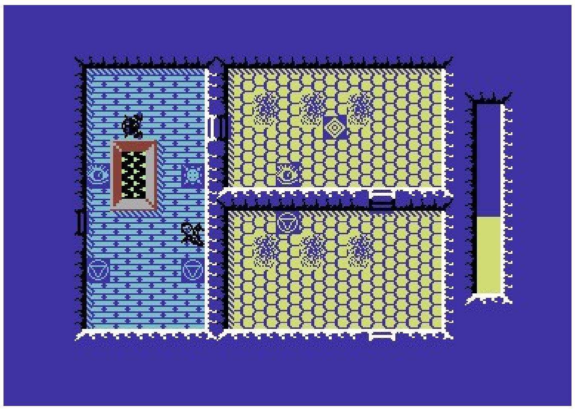
Platform: Commodore 64

Year: 1987

Mervyn, a sorcerer's apprentice was in one of his practice hours with his mentors. He was about to try a new spell for a better look, but this failed and turned him into a frog. While being in this state, evil warlocks appeared and killed all his mentors. He alone was saved, as he appeared only to be a frog. Now Mervyn is the only one left to take revenge on his attackers, by destroying all the warlocks.

So he heads off into the dungeons of the warlocks, consisting of 8 levels. He hasn't lost his magic abilities and can use them to defeat the minions of the warlocks. The warlocks themselves need to be defeated in single fights, by putting together the word "RANARAMA", before his life energy is drained.

ALL POWER EXPIRED
GAME OVER
SCORE 01078400

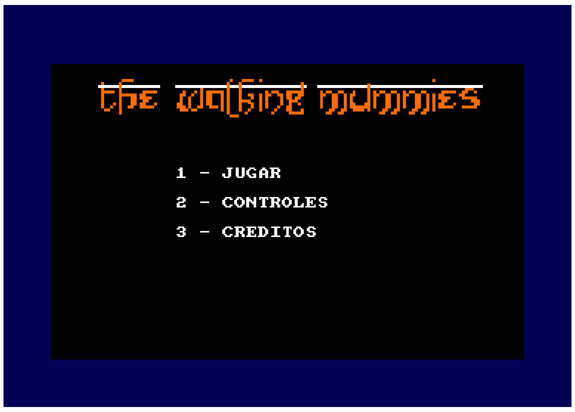


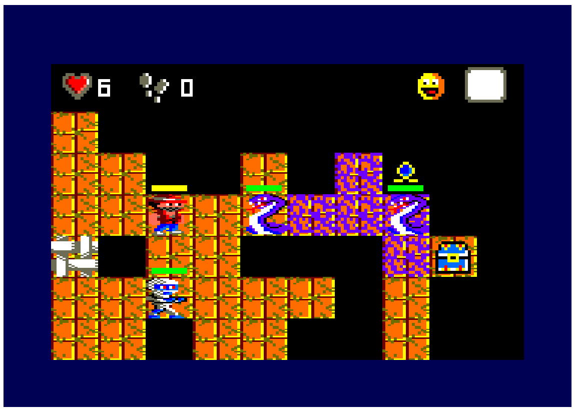
The Walking Mummies[xxii]

Platform: Amstrad CPC

Year: 1985

The Walking Mummies is a smart idea that has been translated into an original and interesting game. You take control of a pyramid explorer who enters mazes with the objective of finding and grabbing treasure chests. This is a strategy game where you take control of the explorer and you need to carefully plan your moves. You get only three steps per round, and after you make your moves, your enemies (cobras and mummies) take their turns. You can fire a limited number of arrows, but they won't help you much.





Mystic Tower[xxiii]

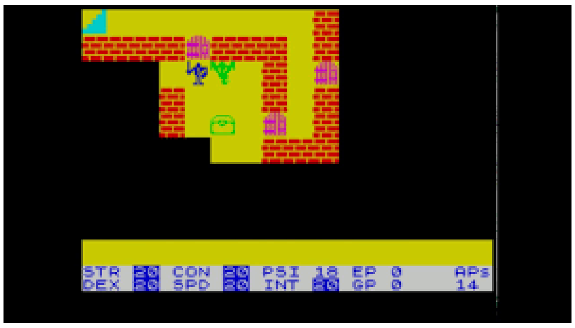
Platform: ZX Spectrum

Year: 1983

Adventure: Dungeon Crawler.

MYSTIC TOWER

IS LOADING
PLEASE WAIT



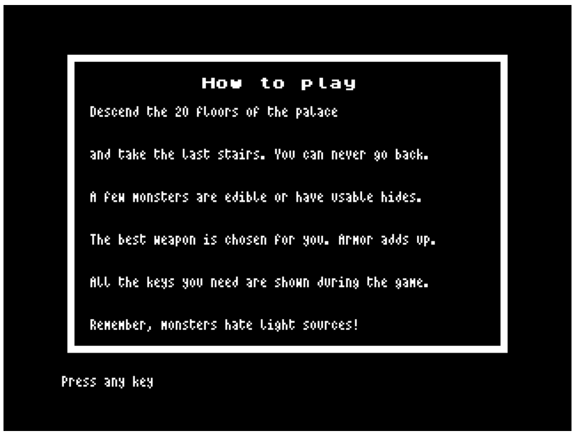
Escape from Cnossus[xxiv]

Platform: ZX Spectrum

Year: 2013

You are the mighty and monstrous Minotaur in Escape from Cnossus. You have spent your years hunting down and slaughtering the many adventurers that dared to enter the palace labyrinth. Now, the king has decided that he no longer has use for you. There will be no more royal and glorious feasts and no more human sacrifices in your honour.

There will be no more hordes of gold and fortune and glory. You've been dismissed, kicked off the royal payroll and left to fend for yourself. Your future employment opportunities look bleak. No one wants to hire an aging Minotaur when the job can be outsourced dirt cheap to menacing fire breathing dragons. Now you long to live out your remaining years in retirement, sitting by a warm fireplace while smoking a pipe and reading a good book. But first you must find your way out of the maze.


How to play

Descend the 20 floors of the palace

and take the last stairs. You can never go back.

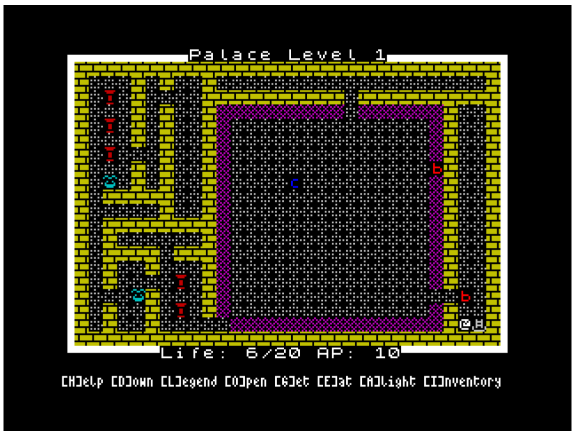
A few monsters are edible or have usable hides.

The best weapon is chosen for you. Armor adds up.

All the keys you need are shown during the game.

Remember, monsters hate light sources!

Press any key


Palace Level 1
Life: 6/20 AP: 10
[H]elp [D]own [L]egend [O]pen [G]et [E]at [A]light [I]nventory

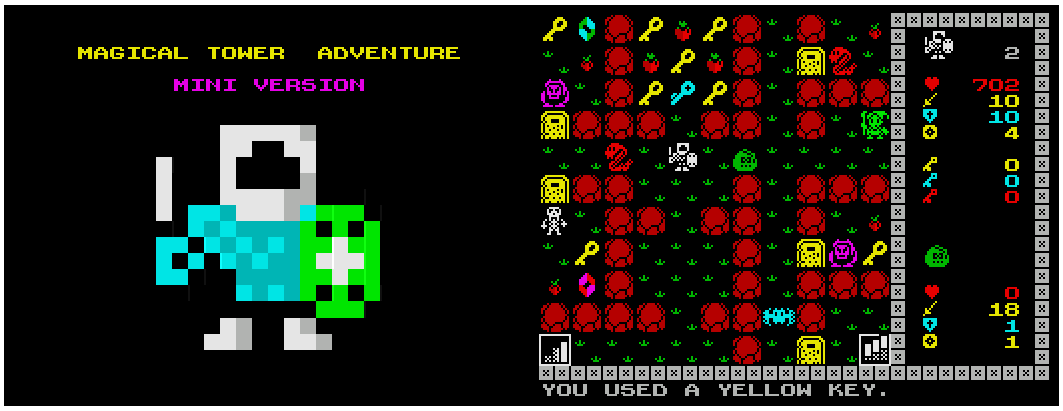
Magical Tower Adventure[xxv]

Platform: ZX Spectrum

Year: 2013

There is a story of a Tower somewhere in the evil lands, filled with monsters and other magical creatures. Many fighters have entered but no one came back. There are many rumours about treasures and princesses, but no one really knows.

This time a new Hero arrived at the Magical Tower, and he will need all the help he can get to kill the boss at the top level of the Tower.



YOU USED A YELLOW KEY.

Time Bandit[xxvi]

Platform: TRS-80, TRS-80 Color Computer, Dragon 32, Atari ST, Amiga, MS-DOS

Year: 1983-1989

Time Bandit is an action-adventure game written for the TRS-80 Model I by Bill Dunlevy and Harry Lafnear and published by MichTron in 1983.[1] It was ported to the TRS-80 Color Computer and Dragon 32, but enjoyed its greatest popularity several years later as an early release for the Atari ST. It was also released for the pseudo-PC-compatible Sanyo MBC-55x with 8-color display. Amiga and MS-DOS versions were ported by Timothy Purves.

Prototype Game Requirements

User Stories

User stories are informal, natural language statements of a software feature from the users perspective[xxvii]. The following user stories are based on the 3Rs, or Connextra format[xxviii]:

As a _____ [role -> persona], I want _____ [requirement -> output], so _____ [reason -> outcome]. The third R (reason) part of this template is optional.

Bill Wake's INVEST mnemonic has also been utilised to create these user stories, or PBI (Product Backlog Items)[xxix]:

Letter	Meaning	Description
I	Independent	The PBI should be self-contained, in a way that there is no inherent dependency on another PBI.
N	Negotiable	PBIs are not explicit contracts and should leave space for discussion.
V	Valuable	A PBI must deliver value to the stakeholders.
E	Estimable	You must always be able to estimate the size of a PBI.
S	Small	PBIs should not be so big as to become impossible to plan/task/prioritize within a level of accuracy.
T	Testable	The PBI or its related description must provide the necessary information to make test development possible.

ID	**As a..**	**I want to..**	**So that..**	**Tasks**	**Time Estimation** (hours)	**Priority** 1 = Very high 2 = High 3 = Medium 4 = Medium to Low 5 = Low
1	Player	Move my character	I can explore the game environment	Input controls		
				Collision detection		
				Render movement		
				Reduce player suite energy (movement based)		
				Reduce player suite oxygen (time based)		
2	Player	Collect objects	I can progress through levels	Object types		
				Player inventory db		
				Inventory menu		
3	Player	Disable 'drone' type non-playing characters	I can retrieve collectibles from them	Collision detection – confirm 'drone'		

Use Case Diagram Example: Player and NPC

The following used case diagram example is a graphic overview of the 'actors' involved in the game and their functions. The actors are the 'player' and the various 'NPCs'. The diagram also shows how the actors functions interact with one another i.e. when the NPC finds the player they 'combat' one another.

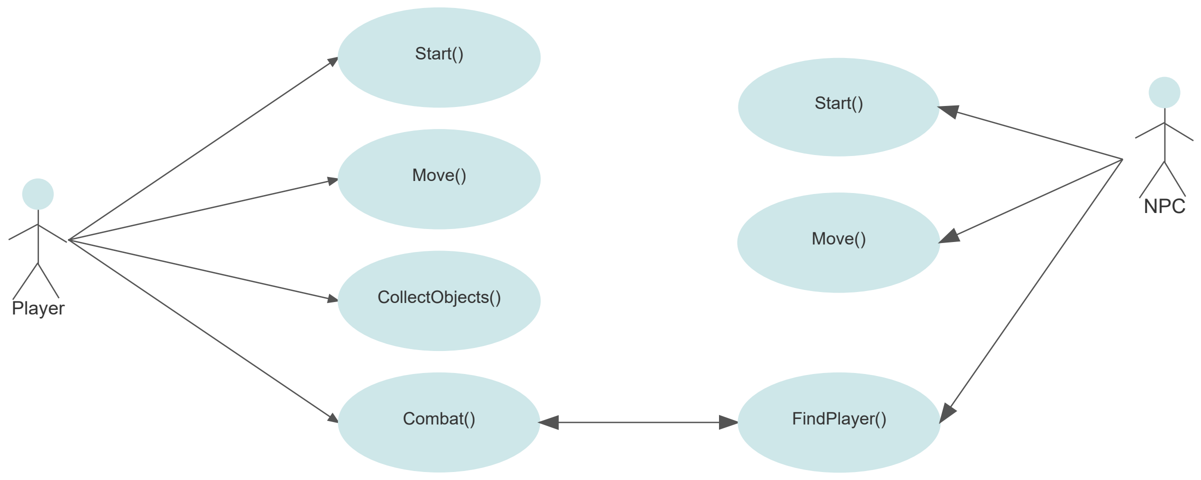


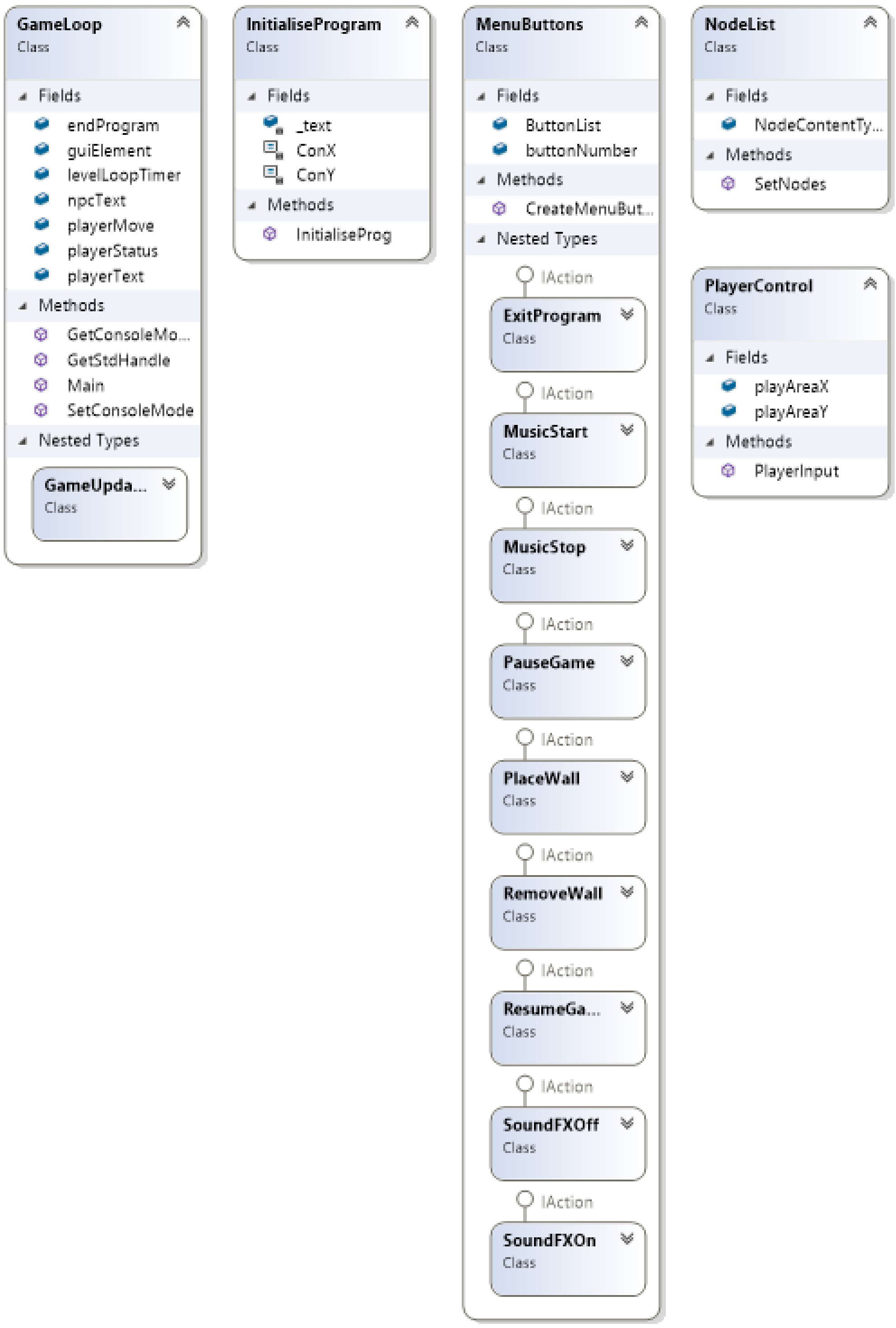
Figure 1. Use Case Diagram Example: Player and NPC

Game Design

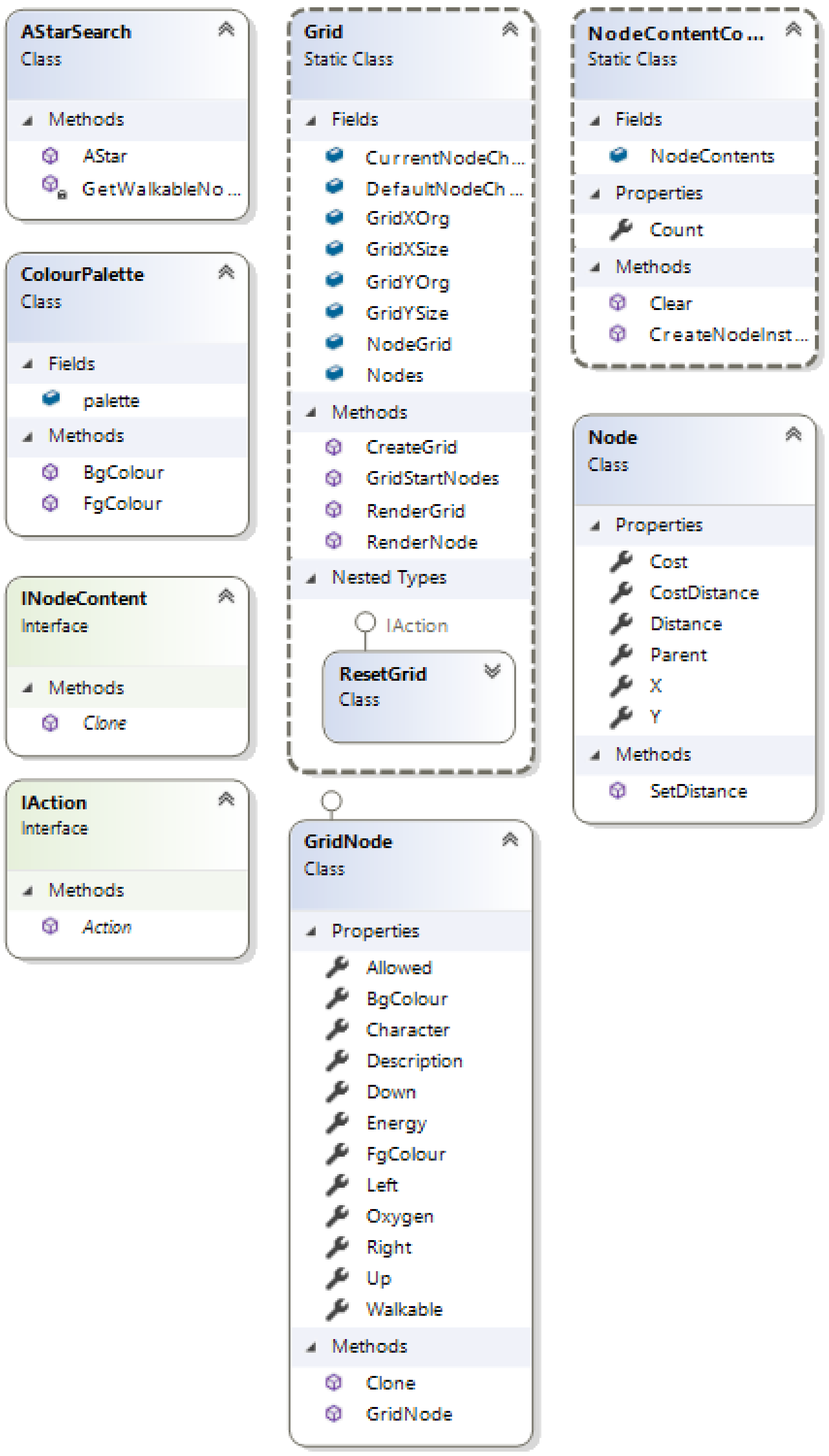
Architectural Design (UML)

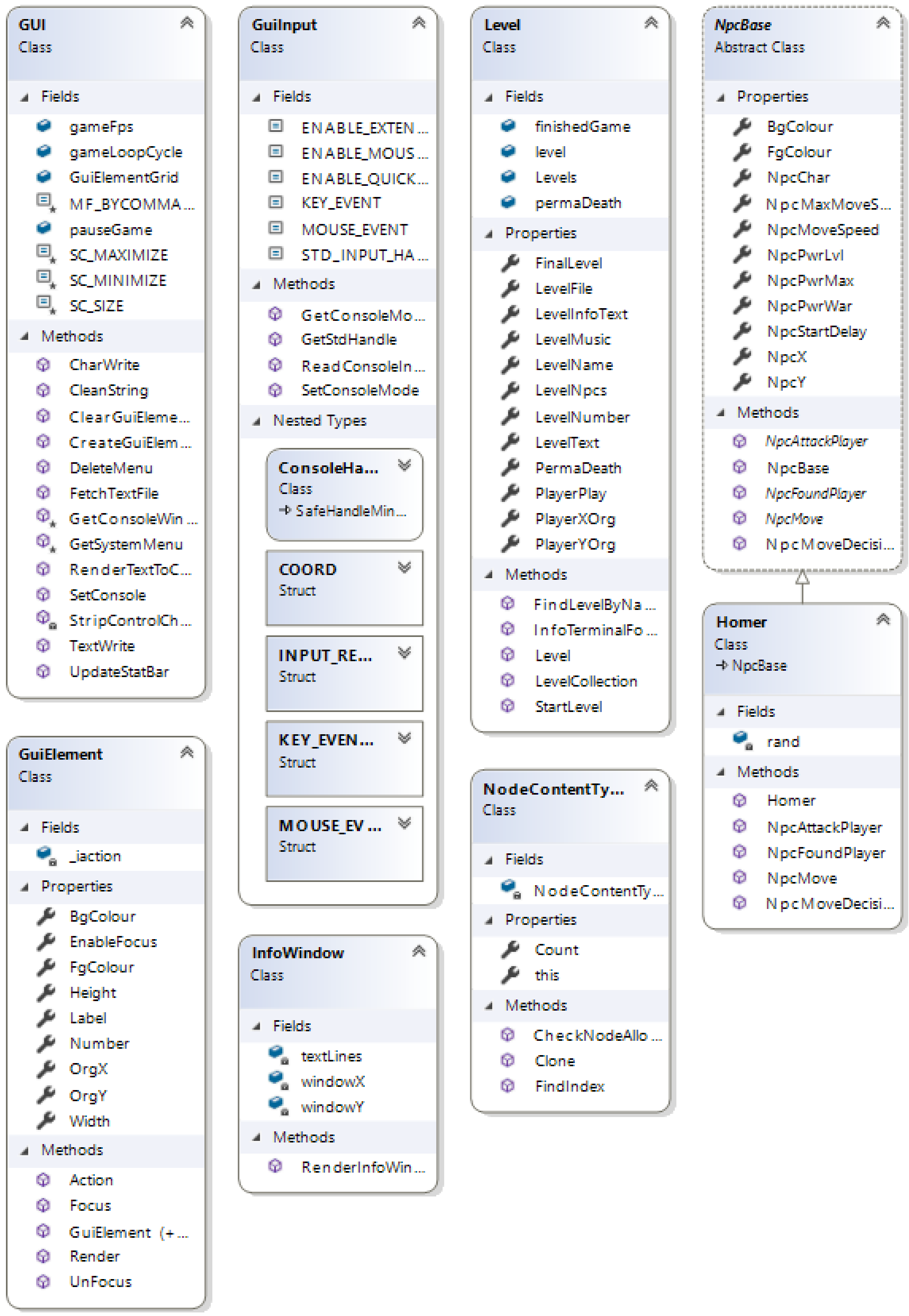
Assessment 2 Solution consists of two areas, Starship One class (start project) and the Class Library.

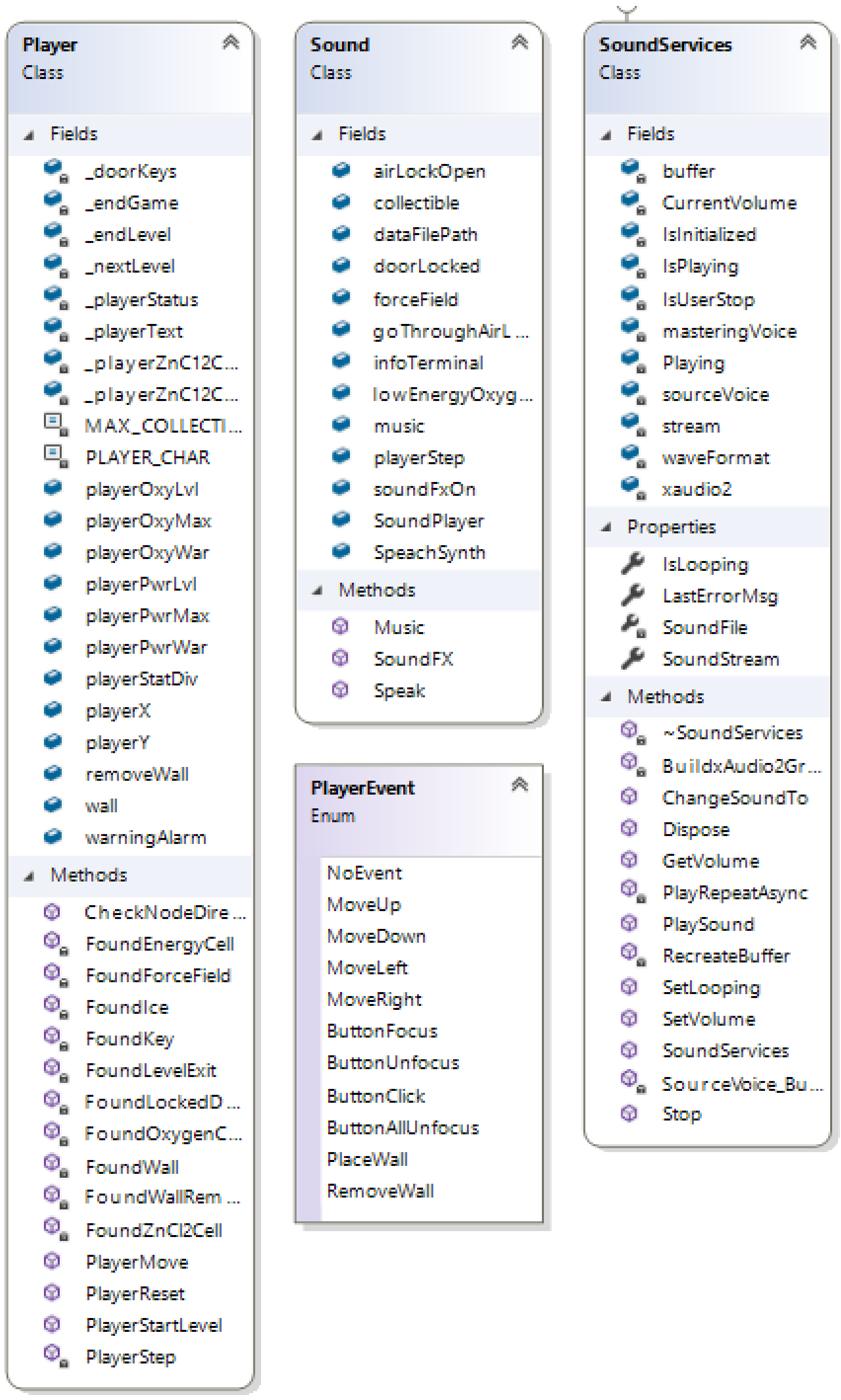
Starship One Class Diagram



Class Library Class Diagram




GUI
Class
Fields
gameFps
gameLoopCycle
GuiElementGrid
MF_BYCOMMA...
pauseGame
SC_MAXIMIZE
SC_MINIMIZE
SC_SIZE
Methods
CharWrite
CleanString
ClearGuiEleme...
CreateGuiElem...
DeleteMenu
FetchTextFile
GetConsoleWin...
GetSystemMenu
RenderTextToC...
SetConsole
StripControlCh...
TextWrite
UpdateStatBar
GuiInput
Class
Fields
ENABLE_EXTEN...
ENABLE_MOUS...
ENABLE_QUICK...
KEY_EVENT
MOUSE_EVENT
STD_INPUT_HA...
Methods
GetConsoleMo...
GetStdHandle
ReadConsoleIn...
SetConsoleMode
Nested Types
ConsoleHa...
Class
SafeHandleMin...
COORD
Struct
INPUT_RE...
Struct
KEY_EVEN...
Struct
MOUSE_EV...
Struct
Level
Class
Fields
finishedGame
level
Levels
permaDeath
Properties
FinalLevel
LevelFile
LevelInfoText
LevelMusic
LevelName
LevelNpcs
LevelNumber
LevelText
PermaDeath
PlayerPlay
PlayerXOrg
PlayerYOrg
Methods
FindLevelByNa...
InfoTerminalFo...
Level
LevelCollection
StartLevel
NpcBase
Abstract Class
Properties
BgColour
FgColour
NpcChar
NpcMaxMoveS...
NpcMoveSpeed
NpcPwrLvl
NpcPwrMax
NpcPwrWar
NpcStartDelay
NpcX
NpcY
Methods
NpcAttackPlayer
NpcBase
NpcFoundPlayer
NpcMove
NpcMoveDecisi...
Homer
Class
NpcBase
Fields
rand
Methods
Homer
NpcAttackPlayer
NpcFoundPlayer
NpcMove
NpcMoveDecisi...
GuiElement
Class
Fields
_iaction
Properties
BgColour
EnableFocus
FgColour
Height
Label
Number
OrgX
OrgY
Width
Methods
Action
Focus
GuiElement (+...
Render
UnFocus
InfoWindow
Class
Fields
textLines
windowX
windowY
Methods
RenderInfoWin...
NodeContentTy...
Class
Fields
NodeContentTy...
Properties
Count
this
Methods
CheckNodeAllo...
Clone
FindIndex


Player
Class
Fields
_doorKeys
_endGame
_endLevel
_nextLevel
_playerStatus
_playerText
_playerZnC12C...
_playerZnC12C...
MAX_COLLECTI...
PLAYER_CHAR
playerOxyLvl
playerOxyMax
playerOxyWar
playerPwrLvl
playerPwrMax
playerPwrWar
playerStatDiv
playerX
playerY
removeWall
wall
warningAlarm
Methods
CheckNodeDire...
FoundEnergyCell
FoundForceField
FoundIce
FoundKey
FoundLevelExit
FoundLockedD...
FoundOxygenC...
FoundWall
FoundWallRem...
FoundZnCl2Cell
PlayerMove
PlayerReset
PlayerStartLevel
PlayerStep
Sound
Class
Fields
airLockOpen
collectible
dataFilePath
doorLocked
forceField
goThroughAirL...
infoTerminal
lowEnergyOxyg...
music
playerStep
soundFxOn
SoundPlayer
SpeachSynth
Methods
Music
SoundFX
Speak
PlayerEvent
Enum
NoEvent
MoveUp
MoveDown
MoveLeft
MoveRight
ButtonFocus
ButtonUnfocus
ButtonClick
ButtonAllUnfocus
PlaceWall
RemoveWall
SoundServices
Class
Fields
buffer
CurrentVolume
IsInitialized
IsPlaying
IsUserStop
masteringVoice
Playing
sourceVoice
stream
waveFormat
xaudio2
Properties
IsLooping
LastErrorMsg
SoundFile
SoundStream
Methods
~SoundServices
BuildxAudio2Gr...
ChangeSoundTo
Dispose
GetVolume
PlayRepeatAsync
PlaySound
RecreateBuffer
SetLooping
SetVolume
SoundServices
SourceVoice_Bu...
Stop

Data Structure Design

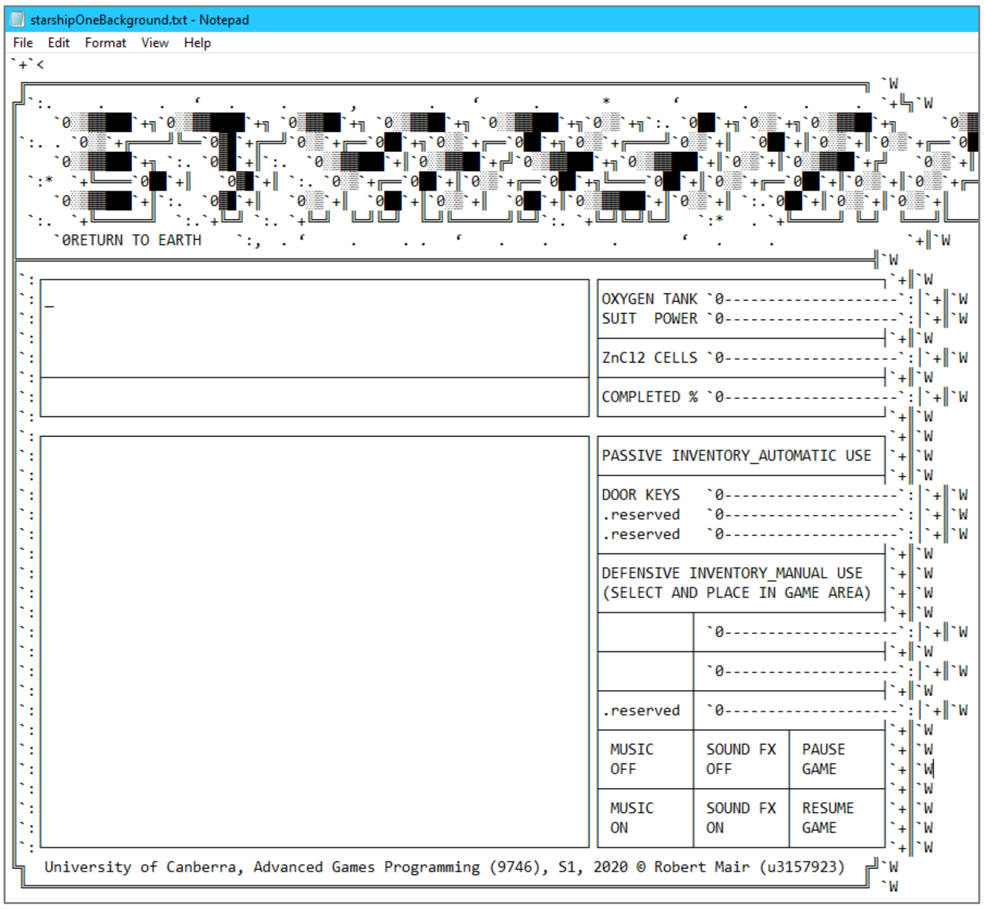
The Starship One prototype game does not use any database (neither SQL or no-SQL) and relies on using .txt files for GUI design/background and level design/play area ('grid') design.

GUI Design/Background

The GUI design/presentation uses basic ASCII with embedded control codes, a tilde ' ` ' followed by the control code per:

• Tilde ' ` ' character then:	*ASCII Code*	*ASCII Chr*
To change foreground colour:	33 to 59	! to ;
To change background colour:	60 to 86	< to V
To complete a 'line return':	87	W
To complete a 'console clear':	88	X

Data file used to create the GUI design/background:


starshipOneBackground.txt - Notepad
File Edit Format View Help

```
`+`<
 `0RETURN TO EARTH
OXYGEN TANK `0-------------------`:
SUIT  POWER `0-------------------`:
ZnCl2 CELLS `0-------------------`:
COMPLETED % `0-------------------`:
PASSIVE INVENTORY_AUTOMATIC USE
DOOR KEYS   `0-------------------`:
.reserved   `0-------------------`:
.reserved   `0-------------------`:
DEFENSIVE INVENTORY_MANUAL USE
(SELECT AND PLACE IN GAME AREA)
            `0-------------------`:
            `0-------------------`:
.reserved   `0-------------------`:
 MUSIC       SOUND FX    PAUSE
 OFF         OFF         GAME
 MUSIC       SOUND FX    RESUME
 ON          ON          GAME
 University of Canberra, Advanced Games Programming (9746), S1, 2020 © Robert Mair (u3157923)
```

Figure 2.GUI design/presentation ASCII data file.

The above file is read into C# and rendered to the Console using the following code:

```
// Read contents of a text file and return it as a string (so it can to be rendere
d to Console Window).
        public static string FetchTextFile(string file)
        {
            string dataFilePath = Directory.GetParent(Directory.GetCurrentDirector
y()).Parent.Parent.FullName +
                                      "\\DataFiles\\" + file;
            if (File.Exists(dataFilePath))
            {
                try
                {
                    string text = File.ReadAllText(dataFilePath);
                    text = StripControlChars(text);
                    return text;
                }
                catch (Exception error)
                {
                    return error.Message;
                }
            }

            return "File Not Found: " + file; ;
        }

        // Clean up string to remove all unwanted characters.
        public static string CleanString(string text)
        {
            text = Regex.Replace(text, @"<[^>]+>|&nbsp;", "").Trim();
            return Regex.Replace(text, @"[^\u0020-\u007F]", String.Empty);
        }

        static string StripControlChars(string arg)
        {
            char[] arrForm = arg.ToCharArray();
            StringBuilder buffer = new StringBuilder(arg.Length); //This many char
s at most

            foreach (char ch in arrForm)
                if (!Char.IsControl(ch)) buffer.Append(ch);//Only add to buffer if
 not a control char

            return buffer.ToString();
        }

        // Render contents of string to Console Window, starting at specified co-
ordinates.
        public static void RenderTextToConsole(int x, int y, string text)
        {
            text = StripControlChars(text);

            Console.SetCursorPosition(x, y);
            bool special = false;
            foreach (char chr in text)
            {
                if (chr == '`')
                {
                    special = true;
                    continue;
                }

                if (special)
                {
                    special = false;
                    int choice = chr - 33;
                    switch (choice)
                    {
                        case int n when n >= 0 && n <= 26:
                            ColourPalette.FgColour(n);
                            break;
                        case int n when n >= 27 && n <= 31:
                            ColourPalette.BgColour(n);
                            break;
                        case 54:
                            y++;
                            Console.SetCursorPosition(x, y);
                            break;
                        case 55:
                            Console.Clear();
                            break;
                    }

                    continue;
                }
                Console.Write(chr);
            }
        }
```

Level design/play area ('grid') design

The level design/play area ('grid') design uses basic ASCII without any embedded control codes, rather each ASCII character represents a 'node' object that is placed into the play area per:

Collectibles

Type	ASCII	Details
ZnCl2 cell	z z	Collect ZnCl2 cells to unlock the section/level.
Energy cell	e e	Increase energy of the player space suite (energy will reduce when disabling automatons or encountering dangerous terrain).
Oxygen cell	o o	Increase oxygen tank level of the player space suite (oxygen will reduce when the player moves).
Tool	t t	Tools that can be used to disable automatons / repair damage.
Space case	c c	A sturdy storage container that can survive the vacuum of space. Don't know what is in a space case, can be a collectible, or booby trap

Terrain

Type	ASCII	Details
Space	- █	Space (outside of the spaceship).
Star		A star in space (outside of the spaceship).
Deck floor	. 🟦	Walkable.
Wall	# 🟧	Walls within the spaceship.
Door	+ +	Between rooms/corridors etc.
Locked door	* *	Between rooms/corridors etc. Must be unlocked to pass through.
Info Terminal	? ?	Player can unlock doors, gain information etc.
Acid	, ,	Walkable, but will reduce player space suite energy.
Ice	~ ~	Unless wearing 'ice' spikes, will cause player to slide in direction they stepped onto the ice.
Force field	_ _ | |	Can block corridors, reduce player space suite energy to walk through (can they be deactivated?
One way (up)	^ ^	Player can move through this in one direction – up.
One way (down)	v v	Player can move through this in one direction – down.
One way (left)	< >	Player can move through this in one direction – left.

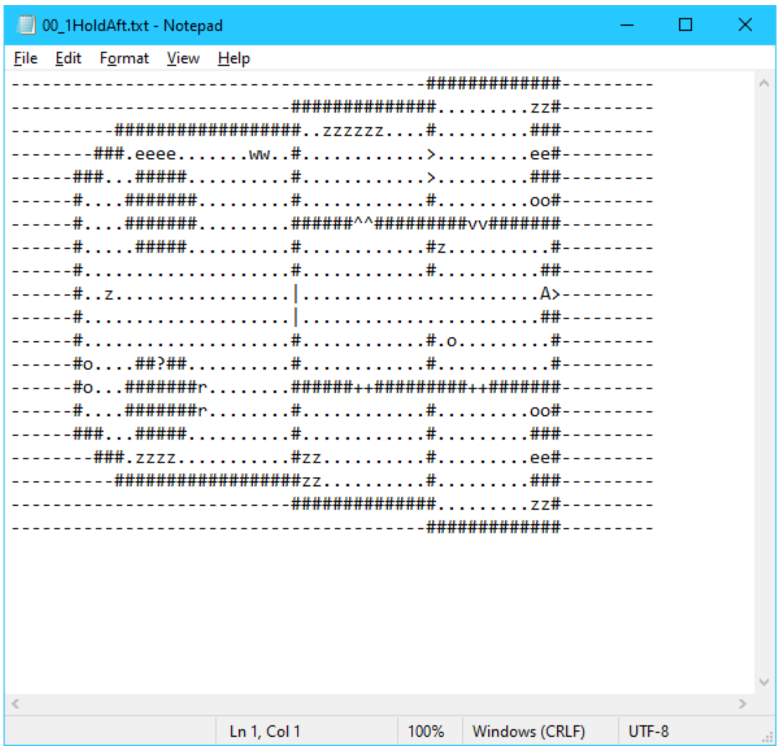
Type	ASCII	Details
One way (right)	> <	Player can move through this in one direction – right.
Airlock (to next section)	A A	Airlock, used to seal each section of a level, operates only once enough ZnCl2 cells are found in the current section.
Lift (to next level)	L L	Lift to next level, operates only once enough ZnCl2 cells are found on the current level.

The above data structure is represented in C# using the following ‘NodeList’ class:

```
using ClassLibrary;

namespace u3157923_9746_Assessment2
{
    public class NodeList
    {
        // Create a list of type 'NodeContentTypes' (use as 'prototypes' for creat
ing 'NodeContentTypes' class instances to place into grid.
        public static NodeContentTypeCollection NodeContentTypes = new NodeContent
TypeCollection();

        public static void SetNodes()
        {

            // Collectible node types.
            NodeContentTypes[NodeContentTypes.Count] = new GridNode('z', "ZnCl2 ce
ll", true, -1, -1, 25, 37, -1, -1, 1, -1, 1);
            NodeContentTypes[NodeContentTypes.Count] = new GridNode('e', "Energy c
ell", true, -1, 49, 25, 37, -1, -1, 1, -1, 1);
            NodeContentTypes[NodeContentTypes.Count] = new GridNode('o', "Oxygen c
ell", true, 199, -1, 25, 37, -1, -1, 1, -1, 1);
            NodeContentTypes[NodeContentTypes.Count] = new GridNode('k', "Key", tr
ue, -1, -1, 25, 37, -1, -1, 1, -1, 1);
            NodeContentTypes[NodeContentTypes.Count] = new GridNode('w', "Temporar
y wall", true, -1, -1, 25, 37, -1, -1, 1, -1, 1);
            NodeContentTypes[NodeContentTypes.Count] = new GridNode('r', "Remove w
all", true, -1, -1, 25, 37, -1, -1, 1, -1, 1);


            // Terrain node types.
            NodeContentTypes[NodeContentTypes.Count] = new GridNode('-
', "space", false, 0, 0, 0, 27, -1, 5, 5, 5, 5);
            NodeContentTypes[NodeContentTypes.Count] = new GridNode('.', "floor",
true, -1, -1, 10, 37, -1, -1, 1, -1, 1);
            NodeContentTypes[NodeContentTypes.Count] = new GridNode('#', "wall", f
alse, 0, 0, 15, 42, -1, 5, 5, 5, 5);
            NodeContentTypes[NodeContentTypes.Count] = new GridNode('W', "Temporar
y wall", false, 0, 0, 0, 42, -1, 5, 5, 5, 5);

            NodeContentTypes[NodeContentTypes.Count] = new GridNode('+', "door", t
rue, -1, -5, 15, 37, -1, -1, 1, -1, 1);
            NodeContentTypes[NodeContentTypes.Count] = new GridNode('*', "locked d
oor", false, -5, -15, 15, 37, -1, -1, 1, -1, 1);
            NodeContentTypes[NodeContentTypes.Count] = new GridNode('?', "info ter
minal", true, -5, -15, 0, 42, -1, -1, 1, -1, 1);
            NodeContentTypes[NodeContentTypes.Count] = new GridNode(',', "acid", t
rue, -5, -50, 15, 37, -1, -1, 1, -1, 1);
```

```
            NodeContentTypes[NodeContentTypes.Count] = new GridNode('~', "ice", tr
ue, -1, -1, 15, 37, -1, -1, 1, -1, 1);
            NodeContentTypes[NodeContentTypes.Count] = new GridNode('_', "force fi
eld", true, -15, -50, 15, 37, -1, -1, 1, -1, 1);
            NodeContentTypes[NodeContentTypes.Count] = new GridNode('|', "force fi
eld", true, -15, -50, 15, 37, -1, -1, 1, -1, 1);

            NodeContentTypes[NodeContentTypes.Count] = new GridNode('^', "door up"
, true, -1, -5, 15, 37, -1, -1, 5, 5, 5);
            NodeContentTypes[NodeContentTypes.Count] = new GridNode('v', "door dow
n", true, -1, -5, 15, 37, -1, 5, 1, 5, 5);
            NodeContentTypes[NodeContentTypes.Count] = new GridNode('<', "door lef
t", true, -1, -5, 15, 37, -1, 5, 5, -1, 5);
            NodeContentTypes[NodeContentTypes.Count] = new GridNode('>', "door rig
ht", true, -1, -5, 15, 37, -1, 5, 5, 5, 1);

            NodeContentTypes[NodeContentTypes.Count] = new GridNode('A', "airlock"
, true, -5, -5, 0, 42, -1, -1, 1, -1, 1);
            NodeContentTypes[NodeContentTypes.Count] = new GridNode('L', "lift", t
rue, -5, -5, 0, 42, -1, -1, 1, -1, 1);

        }
    }
}
```

Data file used to create the GUI design/background:



```
-----------------------------------------#############---------
----------------------------##############.........zz#---------
----------###################..zzzzzz....#.........###---------
--------###.eeee........ww..#............>.........ee#---------
------###...#####...........#............>.........###---------
------#....#######..........#............#.........oo#---------
------#....#######..........######^^#########vv#######---------
------#.....#####...........#............#z..........#---------
------#.....................#............#..........##---------
------#..z..................|.......................A>---------
------#.....................|.......................##---------
------#.....................#............#.o.........#---------
------#o....##?##...........#............#...........#---------
------#o...#######r.........######++#########++#######---------
------#....#######r.........#............#.........oo#---------
------###...#####...........#............#.........###---------
--------###.zzzz............#zz..........#.........ee#---------
----------###################zz..........#.........###---------
----------------------------##############.........zz#---------
-----------------------------------------#############---------
```

Figure 3. GUI design/presentation ASCII data file.

The above file is read into C#, 'node' objects created and then rendered to the Console using the following code:

Reading level data:

```
// If the level is not a 'static level' i.e. the player 'plays' this level, use th
is.
            if (Levels[level].PlayerPlay)
            {
                List<char> gridStartNodes = text.ToList();
                Grid.NodeGrid.Clear();
                Grid.GridStartNodes(gridStartNodes);
                // Setup and render the 'grid' (used for the game play area).
                Grid.CreateGrid();
                InfoWindow.RenderInfoWindow(0); // Clears Info Window.
                foreach (string line in Levels[level].LevelText)
                {
                    InfoWindow.RenderInfoWindow(line, 2);
                }
                Player.PlayerStartLevel(Levels[level].PlayerXOrg,
                    Levels[level].PlayerYOrg);
                if (level > 0) GUI.pauseGame = false;
            }
```

Creating and rendering the level play area ‘grid’ uses the ‘Grid’ class code below:

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace ClassLibrary
{
    public static class Grid
    {

        public static List<char> Nodes;
        public static char DefaultNodeChar; // Default character to place in 'grid
' at each 'node' when first creating the grid.
        public static char CurrentNodeChar; // Currently selected character to pla
ce in 'grid' at each 'node'.
        public static int GridXOrg; // Set X origin of 'grid' (area used for maps/
pathfinding).
        public static int GridYOrg; // Set Y origin of 'grid' (area used for maps/
pathfinding).
        public static int GridXSize; // Set maximum width size of 'grid'.
        public static int GridYSize; // Set maximum height size of 'grid'.

        public static List<List<int>> NodeGrid = new List<List<int>>(); // List of
 lists, to hold 'node' content for the grid.

        public static void GridStartNodes(List<char> newnodes)
        {
            Nodes = newnodes.ToList();
        }

        public static void CreateGrid()
        {
            int nodeNumber = 0;

            for (var i = 0; i < GridYSize; i++)
            {
                NodeGrid.Add(new List<int>());
                for (var j = 0; j < GridXSize; j++) NodeGrid[i].Add(NodeContentCol
lection.CreateNodeInstance(Nodes[nodeNumber++]));
            }
            RenderGrid(GridXOrg, GridYOrg);
        }

        public static void RenderGrid(int xOrg, int yOrg)
        {
            // Example of 'hoisting' for iterating over a 2d list.
            var height = NodeGrid.Count;
            var width = NodeGrid[0].Count;
            for (var y = 0; y < height; y++)
                for (var x = 0; x < width; x++)
                {
                    RenderNode(xOrg, yOrg, x, y);
                    GUI.GuiElementGrid[yOrg + y, xOrg + x] = -1;
                }
        }

        // Render grid node content at node location in grid.
        public static void RenderNode(int xOrg, int yOrg, int x, int y)
        {
            Console.SetCursorPosition(xOrg + x, yOrg + y);
            ColourPalette.FgColour(NodeContentCollection.NodeContents[NodeGrid[y][
x]].FgColour);
            ColourPalette.BgColour(NodeContentCollection.NodeContents[NodeGrid[y][
x]].BgColour);
            Console.Write((char)NodeContentCollection.NodeContents[NodeGrid[y][x]]
.Character);
        }

        // Grid Class generic dependency Injection methods for GuiElement class ob
jects.
        public class ResetGrid : IAction
        {
            public void Action(params object[] list)
            {
                NodeGrid.Clear();
                NodeContentCollection.NodeContents.Clear();
                CreateGrid();
            }
        }

    }
}
```

Level Data

Each ‘level’ in the prototype game is represented by an instance of the ‘Level’ class per the following code (the data also contains which .txt file and sound file is to be used for each level):

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;

namespace ClassLibrary
{
    public class Level
    {
        public static string permaDeath = "Permanent Death!";
        public static string finishedGame = "Game Completed!";
        public static int level = 0;
        public int LevelNumber { get; set; }
        public string LevelName { get; set; }
        public string LevelFile { get; set; }
        public string LevelMusic { get; set; }
        public bool PlayerPlay { get; set; }
        public int PlayerXOrg { get; set; }
        public int PlayerYOrg { get; set; }
        public bool PermaDeath { get; set; }
        public bool FinalLevel { get; set; }
        public List<string> LevelText { get; set; } // Used for text in the 'Info
Window' when each level is started.
        public List<string> LevelInfoText { get; set; } // Used for text in the 'I
nfo Terminal' for when the player uses this to gather information.

        public List<NpcBase> LevelNpcs { get; set; }

        public static List<Level> Levels = new List<Level>();

        // Constructor: for level class instances to be placed into the level coll
ection.

        public Level(int levelNumber, string levelName, string levelFile, string l
evelMusic, bool playerPlay, int playerXOrg, int playerYOrg,
            bool permaDeath, bool finalLevel, List<string> levelText, List<string>
 levelInfoText, List<NpcBase> levelNpcs)
        {
            LevelNumber = levelNumber;
            LevelName = levelName;
            LevelFile = levelFile;
            LevelMusic = levelMusic;
            PlayerPlay = playerPlay;
            PlayerXOrg = playerXOrg;
            PlayerYOrg = playerYOrg;
            PermaDeath = permaDeath;
            FinalLevel = finalLevel;
            LevelText = levelText;
            LevelInfoText = levelInfoText;
            LevelNpcs = levelNpcs;
        }

        public static void LevelCollection()
        {
            Levels.Add(new Level(0, "", "00_0StartScreen.txt", "Underclocked (unde
runderclocked mix).wav",
                false, 0, 0,
                false, false,
                new List<string>
                    { "Press any key to start game (or level number to jump ahead)
." },
                new List<string>
                    { "" },
                new List<NpcBase>()
            ));
            Levels.Add(new Level(0, "00_1 Hold: Aft", "00_1HoldAft.txt", "",
                true, 32, 9, false, false,
                   new List<string>
                   { "Contr.. to Riley ETO, Starship One ..alfunct..launched...",
                       "..check ..info. terminal [?]..for .. more..details......."
},
                   new List<string>
                   {
                       "********** INFORMATION TERMINAL: YOU MUST READ THIS ******
****",
                       "Each level requires 20 'z' ZnC12 cells to be collect. Coll
ect ",
                       "Oxygen 'o' and Energy 'e' cells to maintain your space sui
te! ",
                       //"Cells for your suite to stay alive."
                   },
                   new List<NpcBase>
                       {
                           new Homer('H',50,10,1000,1000,200,15, 27,GUI.gameFps,GU
I.gameFps/1, GUI.gameFps/1),
                           new Homer('H',50,10,1000,1000,200,15, 27,GUI.gameFps*3,
GUI.gameFps/2, GUI.gameFps/2)
                       }
                   ));

            // Player start x=2, y=9
            Levels.Add(new Level(1, "00_2 Hold: Mid", "00_2HoldMid.txt", "",
                true, 57, 9, false, false,
                 new List<string>
                 {
                     "You have made it to level 2, well done Riley...",
                     "..but can you survive and make it to the next one?"
                 },
                 new List<string>
                 {
                     "********** INFORMATION TERMINAL: YOU MUST READ THIS ********
**",
                     "There is ice '~' about, no wonder it is so cold. Keys 'k' ar
e ",
                     "used to open locked doors '*'."
                 },
                 new List<NpcBase>
                 {
                     new Homer('H',57,9,1000,1000,200,15, 27,GUI.gameFps,GUI.gameF
ps/1, GUI.gameFps/1),
                     new Homer('H',57,9,1000,1000,200,15, 27,GUI.gameFps*3, GUI.ga
meFps/2, GUI.gameFps/2),
                     new Homer('H',57,9,1000,1000,200,15, 27,GUI.gameFps*5,GUI.gam
eFps/2, GUI.gameFps/2),
                     new Homer('H',57,9,1000,1000,200,15, 27,GUI.gameFps*8,GUI.gam
eFps/3, GUI.gameFps/3)
                 }
                 ));

            Levels.Add(new Level(2, "04 Level: Navigation_CIC (the final level)",
"04_1NavigationCIC (the final level).txt", "",
                true, 17, 8, false, false,
                new List<string>
                { "You have made it to the Navigation_CIC level.",
                    "This is the last level to complete and",
                    "Starship One will be under your control!" },
                new List<string> { "" },
                new List<NpcBase>
                {
                    new Homer('H',27,10,1000,1000,200,15, 27,GUI.gameFps/2,GUI.gam
eFps/1, GUI.gameFps/1),
                    new Homer('H',45,12,1000,1000,200,15, 27,GUI.gameFps/2,GUI.gam
eFps/1, GUI.gameFps/1),
                    new Homer('H',17,8,1000,1000,200,15, 27,GUI.gameFps*5,GUI.game
Fps/1, GUI.gameFps/1),
                    new Homer('H',17,8,1000,1000,200,15, 27,GUI.gameFps*8, GUI.gam
eFps/3, GUI.gameFps/3),
                    new Homer('H',17,8,1000,1000,200,15, 27,GUI.gameFps*11, GUI.ga
meFps/3, GUI.gameFps/3)

                }
            ));

            Levels.Add(new Level(3, "Game Completed!", "gameCompleted.txt", "",
                false, 0, 0, false, true,
                new List<string>
                { "Well done! You have regained control of Starship One.",
                    "You can now return to Earth as a hero who has saved",
                    "all humankind from a terrible fate! Awesome!" },
                new List<string> { "" },
                new List<NpcBase>()
            ));

            Levels.Add(new Level(4, permaDeath, "permaDeath.txt", "",
                false, 0, 0, true, false,
                new List<string>
                    { "You have suffered a fate worse than death.",
                        "Starship One will now travel to the Andromeda Galaxy.",
                        "All humankind will perish as Earths resources run out!" }
,
                new List<string> { "" },
                new List<NpcBase>()
                ));

        }
```

Data Structure Diagram Example: Level

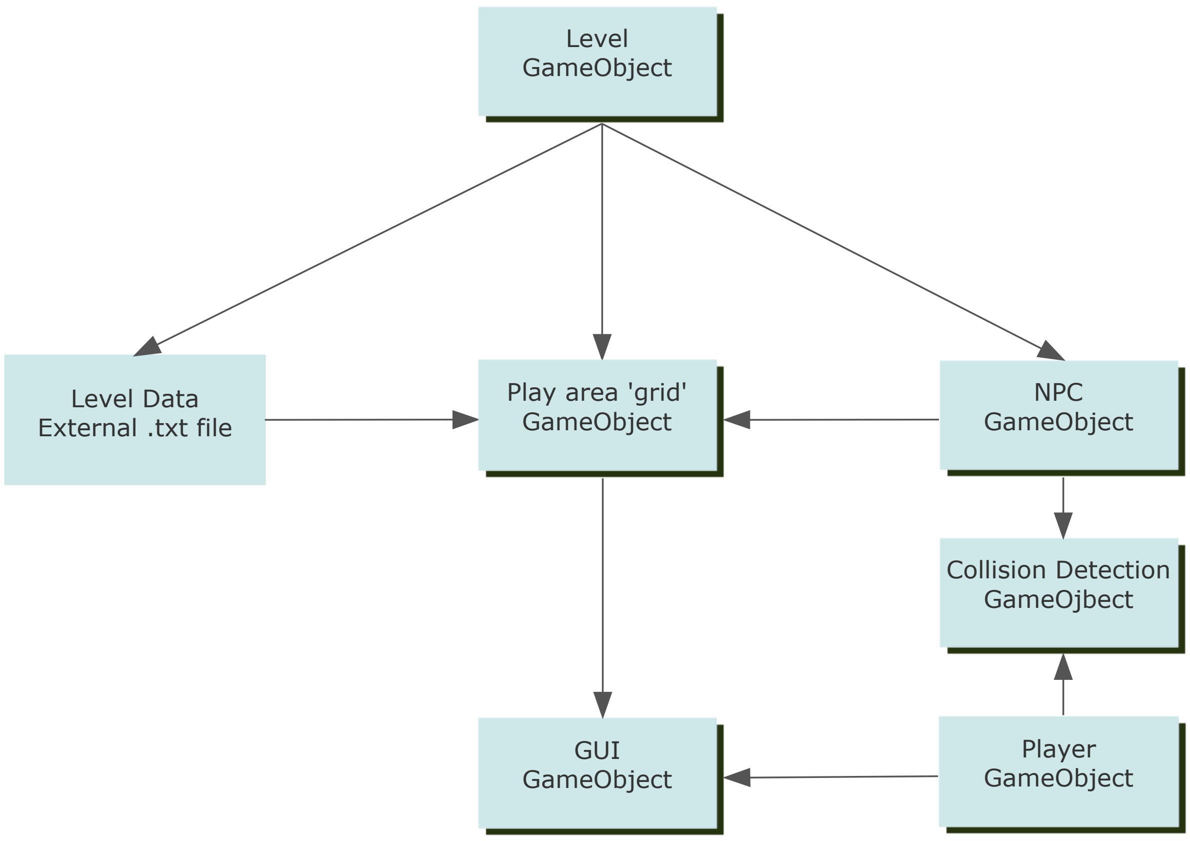

Level GameObject
Level Data External .txt file
Play area 'grid' GameObject
NPC GameObject
Collision Detection GameOjbect
GUI GameObject
Player GameObject

Figure 4. Data Structure Diagram for Level GameObject

Entity-Relationship Diagrams

The following Entity-Relationship Diagram (ERD) shows a high-level representation of the interconnectedness of key components of (entities) of the Starship One project. A detailed listing of the attributes of each entity is outlined in the UML class design diagrams.

ERD's are used to help describe interrelated 'things of interest' within a particular knowledge domain. Basic ERD models comprises of entity types (these highlight things of interest) and the relationships between these entity types.

ERD's are an abstract data model that are typically used to define a data structure for a database (relational database typically) and where developed by Peter Chen and published in 1976.^{xxx}

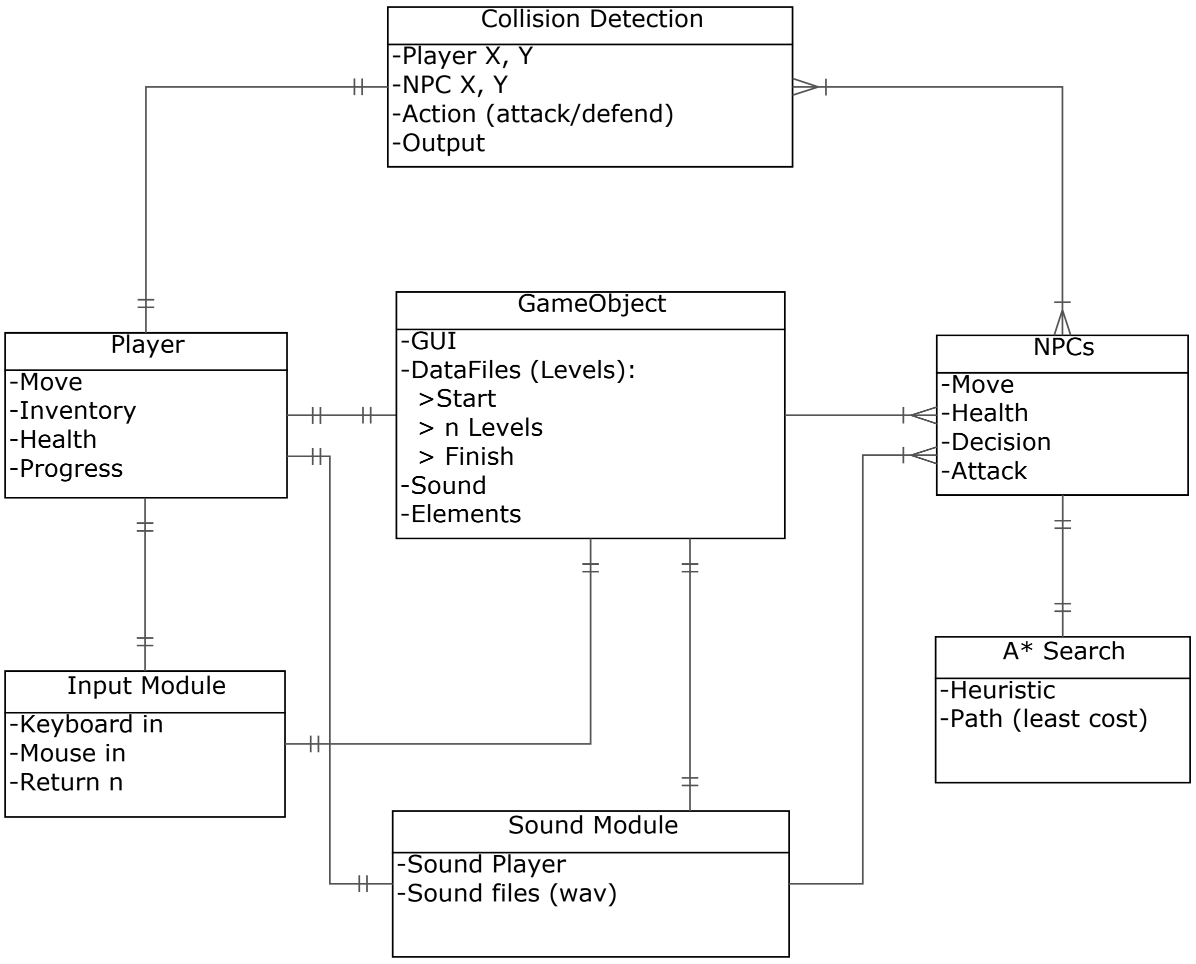


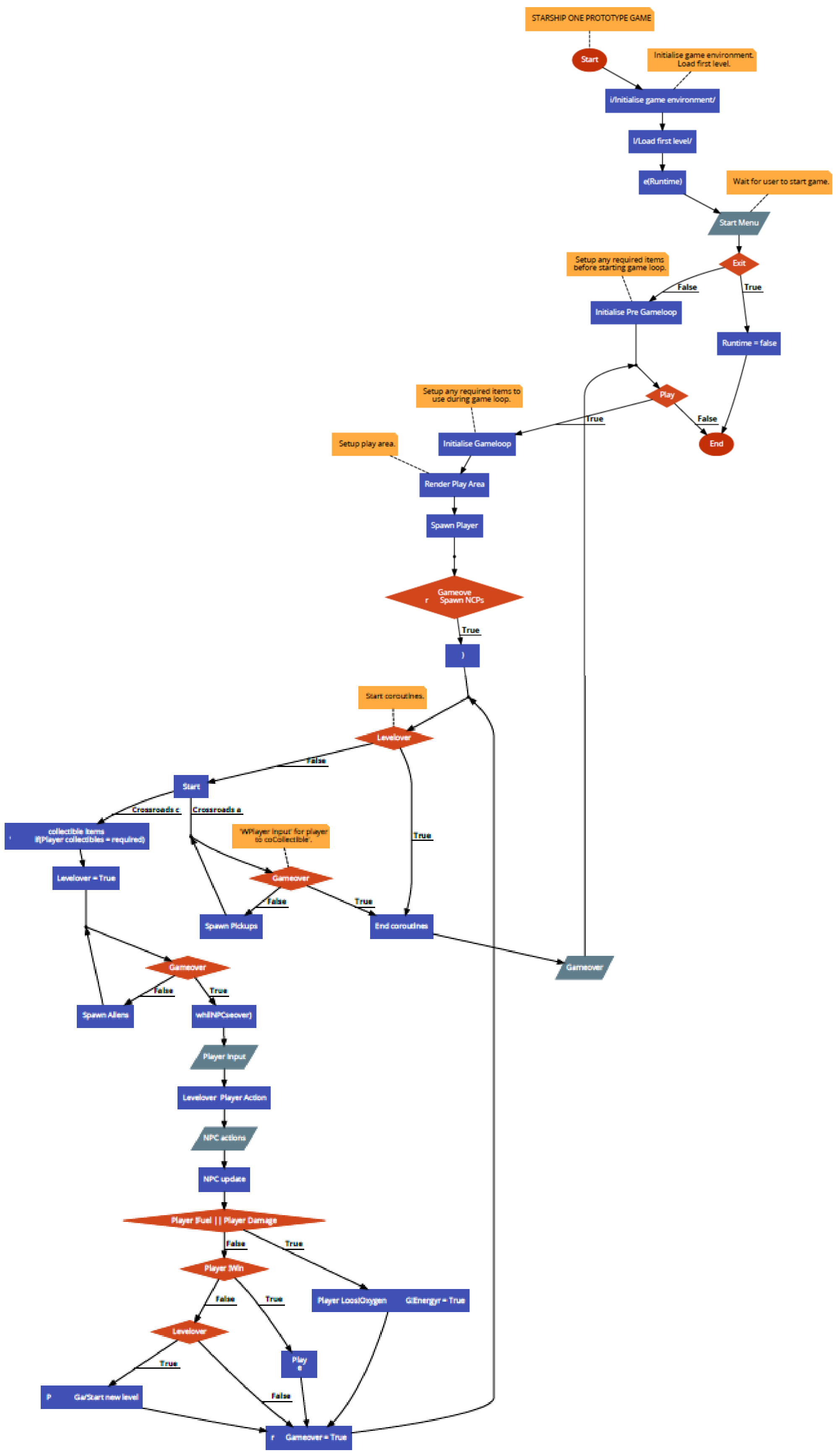
Figure 5. High level Entity Relationship Diagram for Starship One.

Algorithm Design – Pseudocode – high level

```
// STARSHIP ONE PROTOTYPE GAME
Start;
// Initialise game environment.
// Load first level.
i/Initialise game environment/;
l/Load first level/;e(Runtime)
{
 // Wait for user to start game.
 /Start Menu/;
 if(!Exit) {
  // Setup any required items before starting game loop.
  Initialise Pre Gameloop;
  while(Play)
  {
   // Setup any required items to use during game loop.
   Initialise Gameloop;
   // Setup play area.
   Render Play Area;
   Spawn Player;
   while(!Gameove
r     Spawn NCPs;)
   {
    // Start coroutines.
   while(!Levelover)
   {
      Start [Crossroads c];
    // 'WPlayer input' for player to coCollectible'.
    branch(cross_a) [Crossroads a]
    {
     while(!Gameover)
     {
      Spawn Pickups;
     }
```

```
        }
   collectible items
'            if(Player collectibles = required) {Levelover = True}
        {
          while(!Gameover)
          {
            Spawn Aliens;
          }
        }
        whilNPCseover)
        {
          /Player Input/;
Levelover  Player Action;
        /NPC actions/;
          NPC update;
          if(Player !Fuel || Player Damage) {
            Player Loos!Oxygen          G!Energyr = True;
          } else if(Player !Win) {
            Play
e           } else if(Levelover) {
            P             Ga/Start new level          }r      Gameover = True;
          }
        }
        join(cross_a);
        End coroutines;
      }
      /Gameover/;
    }
  } else {
    Runtime = false;
  }
}
```

Algorithm Design - high level – DONE



Game Implementation

Code with Comments

Example of code from the Starship One project (all code can be viewed in the Visual Studio solution):

ClassLibrary > Player

```
namespace ClassLibrary
{
    public class Player
    {
        private const char PLAYER_CHAR = '@';
        public static int playerX;
        public static int playerY;
        public static int playerOxyMax = 1000;
        public static int playerOxyLvl = 1000;
        public static int playerOxyWar = 250;
        public static int playerPwrMax = 1000;
        public static int playerPwrLvl = 1000;
        public static int playerPwrWar = 250;
        public static int playerStatDiv = 50; // Divides oxygen and power values t
o align with stat bars on GUI as they only have 20 marks (so 1000/50=20).

        private static int _playerZnC12CellsLvl;
        private static int _playerZnC12CellsMax = 1;
        private static bool _endLevel;
        private static bool _nextLevel;
        private static bool _endGame = false;
        private static int _playerStatus = -
1; // Return code to main program loop (e.g. level completed, out of oxygen or ene
rgy).
        private static string _playerText;
        public static bool warningAlarm;

        // Player 'collectibles' to start the game.
        private static int _doorKeys;
        public static int wall = 10; // Inventory of 'temporary walls' the player
can place in the play area to defend against NPCs.
        public static int removeWall = 10; // Inventory of ability to remove 'temp
orary walls'. Handy in case the player has blocked an exit.
        private const int MAX_COLLECTIBLE = 20;

        public static void PlayerReset()
        {
            playerOxyLvl = playerOxyMax;
            playerPwrLvl = playerPwrMax;
            _playerZnC12CellsLvl = 0;
            _endLevel = false;
            _nextLevel = false;
            _playerStatus = -1;
            _doorKeys = 0;
            wall = 10;
        }
        public static void PlayerStartLevel(int playerXOrg, int playerYOrg)
        {
            _endLevel = false;
            _nextLevel = false;
            playerX = playerXOrg;
            playerY = playerYOrg;
            _playerZnC12CellsLvl = 0;
            _playerStatus = -1;
            GUI.UpdateStatBar(80, 13, playerOxyMax / playerStatDiv, playerOxyLvl /
 playerStatDiv, playerOxyWar / playerStatDiv);
            GUI.UpdateStatBar(80, 14, playerPwrMax / playerStatDiv, playerPwrLvl /
 playerStatDiv, playerPwrWar / playerStatDiv);
            GUI.UpdateStatBar(80, 16, 20, _playerZnC12CellsLvl, 0);
            GUI.CharWrite(Grid.GridXOrg + playerX, Grid.GridYOrg + playerY, PLAYER
_CHAR, 25, 27);
        }

        public static (string, int) PlayerMove(int moveX, int moveY)
        {
            _playerStatus = -1;
            _playerText = "";

            char nextNode = CheckNodeDirection(playerX, playerY, moveX, moveY);
            if (playerX + moveX > 0 &&
                playerX + moveX < Grid.GridXSize &&
                playerY + moveY > 0 &&
                playerY + moveY < Grid.GridYSize &&
                NodeContentCollection.NodeContents[Grid.NodeGrid[playerY + moveY][
playerX + moveX]].Walkable &&
                nextNode != '`' || nextNode == '*')
            {

                if (nextNode == '*')
                {
                    FoundLockedDoor(moveX, moveY);
                    return (_playerText, _playerStatus);
                }

                Grid.RenderNode(Grid.GridXOrg, Grid.GridYOrg, playerX, playerY);
// Renders grid node content for where player is moving from.
                playerX += moveX;
                playerY += moveY;
                GUI.CharWrite(Grid.GridXOrg + playerX, Grid.GridYOrg + playerY, PL
AYER_CHAR, 25, 27); // Renders player in new location.

                PlayerStep(); // Updates oxygen and energy level of player for eac
h move they make.

                // Checks for what the player is going to find on the next node th
ey step onto, and what to do if they find these items.
                if (nextNode == 'z') FoundZnCl2Cell();
                if (nextNode == 'e') FoundEnergyCell();
                if (nextNode == 'o') FoundOxygenCell();
                if (nextNode == '-' || nextNode == '|') FoundForceField();
                if (nextNode == 'A' || nextNode == 'L') FoundLevelExit(nextNode);
                if (nextNode == '~')
                {
                    GUI.pauseGame = true;
                    FoundIce(moveX, moveY);
                }

                if (nextNode == '?') _playerStatus = 3;
                if (nextNode == 'k') FoundKey();
                if (nextNode == 'w') FoundWall();
                if (nextNode == 'r') FoundWallRemover();

            }

            if (_nextLevel) // Player has completed the level, time to go to the n
ext level.
            {
                _playerStatus = 0;
                _nextLevel = false;
            }

            if (_endGame) // Player has completed the game.
            {
                _playerStatus = 4;
                _nextLevel = false;
                _endGame = false;
            }

            if (playerOxyLvl <= 0) _playerStatus = 1; // Player is out of oxygen,
permadeath.
            if (playerPwrLvl <= 0) _playerStatus = 2; // Player is out of energy,
permadeath.

            return (_playerText, _playerStatus);
        }

        // Nodes are 'directional' e.g. you can only step into a node from certain
 directions, this function checks to ensure the direction
        // the node is being entered from is allowable.
        public static char CheckNodeDirection(int originX, int originY, int moveX,
 int moveY)
        {
            if (NodeContentCollection.NodeContents[Grid.NodeGrid[originY + moveY][
originX + moveX]].Up == moveY ||
                NodeContentCollection.NodeContents[Grid.NodeGrid[originY + moveY][
originX + moveX]].Down == moveY ||
                NodeContentCollection.NodeContents[Grid.NodeGrid[originY + moveY][
originX + moveX]].Left == moveX ||
                NodeContentCollection.NodeContents[Grid.NodeGrid[originY + moveY][
originX + moveX]].Right == moveX
            )
            {
                return NodeContentCollection.NodeContents[Grid.NodeGrid[originY +
moveY][originX + moveX]].Character;
            }

            return '`';
        }

        private static void PlayerStep()
        {
            Sound.playerStep.SetVolume(0.5f);
            if (Sound.soundFxOn) Sound.playerStep.PlaySound();
            // Every move player makes changes oxygen level.
            playerOxyLvl += NodeContentCollection.NodeContents[Grid.NodeGrid[playe
rY][playerX]].Oxygen;
            if (playerOxyLvl < 0) playerOxyLvl = 0;
            if (playerOxyLvl > playerOxyMax) playerOxyLvl = playerOxyMax;
            // Every move player makes changes space suite power level.
            playerPwrLvl += NodeContentCollection.NodeContents[Grid.NodeGrid[playe
rY][playerX]].Energy;
            if (playerPwrLvl < 0) playerPwrLvl = 0;
            if (playerPwrLvl > playerPwrMax) playerPwrLvl = playerOxyMax;


        }

        private static void FoundZnCl2Cell()
        {

            if (_playerZnC12CellsLvl < _playerZnC12CellsMax)
            {
                if (Sound.soundFxOn) Sound.collectible.PlaySound();
                _playerZnC12CellsLvl += 1;
                GUI.UpdateStatBar(80, 16, 20, _playerZnC12CellsLvl, 0);
                _playerText = _playerZnC12CellsLvl == 1 ? "Well done, your first Z
nCl2 cell!" : "That's ZnCl2 cell number " + _playerZnC12CellsLvl + ", " + (_player
ZnC12CellsMax - _playerZnC12CellsLvl) + " to go.";

            }

            if (_playerZnC12CellsLvl == _playerZnC12CellsMax && _endLevel == false
)
            {
                if (Sound.soundFxOn) Sound.airLockOpen.PlaySound();
                _playerText = "All ZnCl2 cells collected, airlock has opened.";
                _endLevel = true;

            }
            Grid.NodeGrid[playerY][playerX] = NodeContentCollection.CreateNodeInst
ance('.');
        }

        private static void FoundOxygenCell()
        {
            if (Sound.soundFxOn) Sound.collectible.PlaySound();
            _playerText = playerOxyLvl == playerOxyMax
                ? "An oxygen cell, maximum oxygen level restored!"
                : "An oxygen cell, you have some more breathing space!";
            Grid.NodeGrid[playerY][playerX] = NodeContentCollection.CreateNodeInst
ance('.');
        }

        private static void FoundEnergyCell()
        {
            if (Sound.soundFxOn) Sound.collectible.PlaySound();
            _playerText = playerPwrLvl == playerPwrMax
                ? "An energy cell, maximum suite energy restored!"
                : "An energy cell, feeling the power?";
            Grid.NodeGrid[playerY][playerX] = NodeContentCollection.CreateNodeInst
ance('.');
        }

        private static void FoundForceField()
        {
            if (Sound.soundFxOn) Sound.forceField.PlaySound();
            _playerText = playerPwrLvl == 0
                ? "A force field, it has sapped all your suite power!"
                : "A force field, it has sapped some of your suite power!";
        }

        private static void FoundLevelExit(char nextNode)
        {
            if (_endLevel)
            {
                if (Sound.soundFxOn) Sound.goThroughAirLock.PlaySound();
                _nextLevel = true;
                if (Level.level == 3) _endGame = true;
            }
            else
            {
                if (Sound.soundFxOn) Sound.doorLocked.PlaySound();
            }

            string exitType = nextNode == 'A' ? "AIRLOCK" : "LIFT";
            _playerText = _endLevel
                ? "The " + exitType + " is now unlocked, get ready for the next le
vel!"
                : "The " + exitType + " is locked, you haven't found all the ZnCl2
 cells.";
        }

        private static void FoundIce(int moveX, int moveY)
        {
            _playerText = "Sliding on ice...";
            GUI.pauseGame = false;
            PlayerMove(moveX, moveY);
        }

        private static void FoundKey()
        {
            if (Sound.soundFxOn) Sound.collectible.PlaySound();
            if (_doorKeys < MAX_COLLECTIBLE) _doorKeys++;
            _playerText = "You have found a door key, well done.";
            GUI.UpdateStatBar(80, 23, MAX_COLLECTIBLE, _doorKeys, 0);
            Grid.NodeGrid[playerY][playerX] = NodeContentCollection.CreateNodeInst
ance('.');
        }

        private static void FoundLockedDoor(int moveX, int moveY)
        {
            if (_doorKeys > 0)
            {
                _playerText = "You have unlocked the door.";
                _doorKeys--;
                Grid.NodeGrid[playerY + moveY][playerX + moveX] = NodeContentColle
ction.CreateNodeInstance('+');
                Grid.RenderNode(Grid.GridXOrg, Grid.GridYOrg, playerX + moveX, pla
yerY + moveY);
                GUI.UpdateStatBar(80, 23, MAX_COLLECTIBLE, _doorKeys, 0);
            }
            else
            {
                _playerText = "The door is locked, you need a key!";
            }
        }

        private static void FoundWall()
        {
            if (Sound.soundFxOn) Sound.collectible.PlaySound();
            if (wall < 20) wall++;
            Grid.NodeGrid[playerY][playerX] = NodeContentCollection.CreateNodeInst
ance('.');
        }

        private static void FoundWallRemover()
        {
            if (Sound.soundFxOn) Sound.collectible.PlaySound();
            if (removeWall < 20) removeWall++;
            Grid.NodeGrid[playerY][playerX] = NodeContentCollection.CreateNodeInst
ance('.');
        }

    }
}
```

u3157923_9746_Assessment2 (Starship One) > MenuButtons

```
using ClassLibrary;
using System;
using System.Collections.Generic;

namespace u3157923_9746_Assessment2
{
    public class MenuButtons
    {
        public static readonly List<GuiElement> ButtonList = new List<GuiElement>(
);
        public static int buttonNumber = -
1; // Used to hold a value of any GUI element 'button' that has come into 'focus'
so it can be 'unfocused'.

        public static void CreateMenuButtons()
        {

            // Game Menu Buttons: Player inventory/actions.

            ButtonList.Add(new GuiElement(69, 30, 8, 1, " WALL", 25, 42, ButtonLis
t.Count + 1, true, new PlaceWall()));
            ButtonList.Add(new GuiElement(69, 32, 8, 1, " REMOVE", 25, 42, ButtonL
ist.Count + 1, true, new RemoveWall()));

            // Game Menu Buttons: In game options.

            ButtonList.Add(new GuiElement(68, 36, 10, 2, " MUSIC` ON", 25, 42, But
tonList.Count + 1, true, new MusicStart()));
            ButtonList.Add(
                new GuiElement(79, 36, 10, 2, " SOUND FX` ON", 25, 42, ButtonList.
Count + 1, true, new SoundFXOn()));
            ButtonList.Add(
                new GuiElement(90, 36, 10, 2, " RESUME` GAME", 25, 42, ButtonList.
Count + 1, true, new ResumeGame()));
```

```
            ButtonList.Add(new GuiElement(68, 39, 10, 2, " MUSIC` OFF", 25, 42, Bu
ttonList.Count + 1, true, new MusicStop()));
            ButtonList.Add(new GuiElement(79, 39, 10, 2, " SOUND FX` OFF", 25, 42,
 ButtonList.Count + 1, true,
                new SoundFXOff()));
            ButtonList.Add(new GuiElement(90, 39, 10, 2, " PAUSE` GAME", 25, 42, B
uttonList.Count + 1, true,
                new PauseGame()));

            ButtonList.Add(new GuiElement(89, 10, 8, 1, "  QUIT", 25, 42, ButtonLi
st.Count + 1, true, new ExitProgram()));
        }

        // This project dependency Injection methods for GuiElement class objects.


        public class MusicStart : IAction
        {
            public void Action(params object[] list)
            {
                // Sound.SoundPlayer.PlayLooping();
                Sound.music.PlaySound();
            }
        }
        public class MusicStop : IAction
        {
            public void Action(params object[] list)
            {
                // Sound.SoundPlayer.Stop();
                Sound.music.Stop();
            }
        }

        public class SoundFXOn : IAction
        {
            public void Action(params object[] list)
            {
                Sound.soundFxOn = true;
            }
        }

        public class SoundFXOff : IAction
        {
            public void Action(params object[] list)
            {
                Sound.soundFxOn = false;
            }
        }

        public class PlaceWall : IAction
        {
            public void Action(params object[] list)
            {
                Grid.CurrentNodeChar = 'W';
                GUI.TextWrite(68, 30, ">", 25, 27, 0);
                GUI.TextWrite(68, 32, " ", 25, 27, 0);
            }
        }
        public class RemoveWall : IAction
        {
            public void Action(params object[] list)
            {
                Grid.CurrentNodeChar = 'r';
                GUI.TextWrite(68, 30, " ", 25, 27, 0);
                GUI.TextWrite(68, 32, ">", 25, 27, 0);
            }
        }

        public class PauseGame : IAction
        {
            public void Action(params object[] list)
            {

                GUI.pauseGame = true;
            }
        }
        public class ResumeGame : IAction
        {
            public void Action(params object[] list)
            {

                GUI.pauseGame = false;
            }
        }
        public class ExitProgram : IAction
        {
            public void Action(params object[] list)
            {
                GameLoop.endProgram = true;
                Console.Clear();
                Console.Write("\n Thank you for using this program, press any key 
to fully exit...");
                Console.ReadKey();
            }
        }

    }
}
```

ClassLibrary > Grid

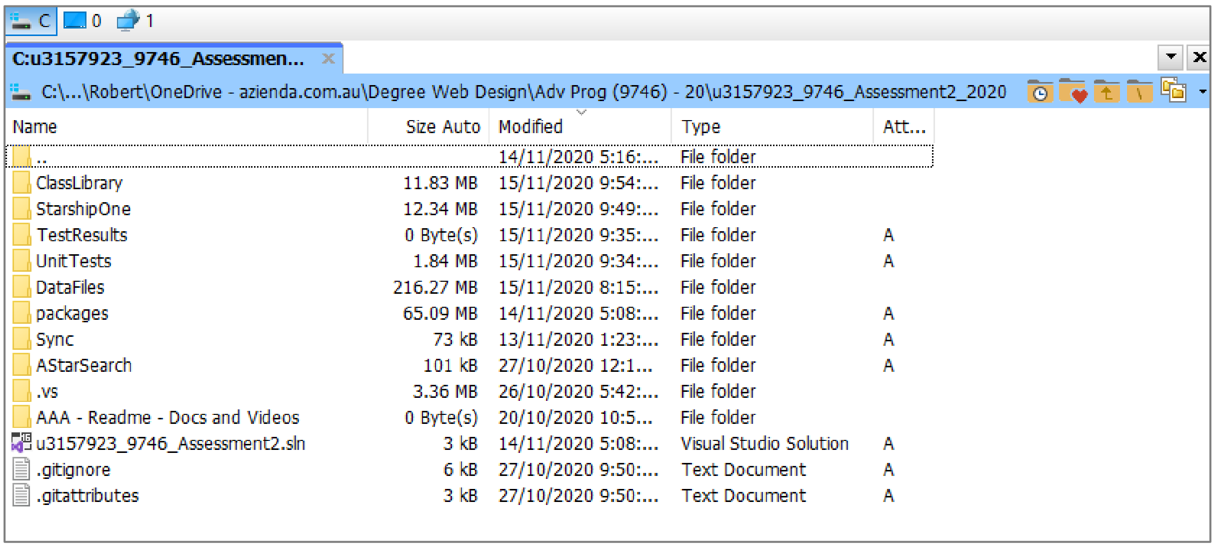
```
using System;
using System.Collections.Generic;
using System.Linq;

namespace ClassLibrary
{
    public static class Grid
    {

        public static List<char> Nodes;
        public static char DefaultNodeChar; // Default character to place in 'grid
' at each 'node' when first creating the grid.
```

```
        public static char CurrentNodeChar; // Currently selected character to pla
ce in 'grid' at each 'node'.
        public static int GridXOrg; // Set X origin of 'grid' (area used for maps/
pathfinding).
        public static int GridYOrg; // Set Y origin of 'grid' (area used for maps/
pathfinding).
        public static int GridXSize; // Set maximum width size of 'grid'.
        public static int GridYSize; // Set maximum height size of 'grid'.

        public static List<List<int>> NodeGrid = new List<List<int>>(); // List of
 lists, to hold 'node' content for the grid.

        public static void GridStartNodes(List<char> newnodes)
        {
            Nodes = newnodes.ToList();
        }

        public static void CreateGrid()
        {
            int nodeNumber = 0;

            for (var i = 0; i < GridYSize; i++)
            {
                NodeGrid.Add(new List<int>());
                for (var j = 0; j < GridXSize; j++) NodeGrid[i].Add(NodeContentCol
lection.CreateNodeInstance(Nodes[nodeNumber++]));
            }
            RenderGrid(GridXOrg, GridYOrg);
        }

        public static void RenderGrid(int xOrg, int yOrg)
        {
            // Example of 'hoisting' for iterating over a 2d list.
            var height = NodeGrid.Count;
            var width = NodeGrid[0].Count;
            for (var y = 0; y < height; y++)
                for (var x = 0; x < width; x++)
                {
                    RenderNode(xOrg, yOrg, x, y);
                    GUI.GuiElementGrid[yOrg + y, xOrg + x] = -1;
                }
        }

        // Render grid node content at node location in grid.
        public static void RenderNode(int xOrg, int yOrg, int x, int y)
        {
            Console.SetCursorPosition(xOrg + x, yOrg + y);
            ColourPalette.FgColour(NodeContentCollection.NodeContents[NodeGrid[y][
x]].FgColour);
            ColourPalette.BgColour(NodeContentCollection.NodeContents[NodeGrid[y][
x]].BgColour);
            Console.Write((char)NodeContentCollection.NodeContents[NodeGrid[y][x]]
.Character);
        }

        // Grid Class generic dependency Injection methods for GuiElement class ob
jects.
        public class ResetGrid : IAction
        {
            public void Action(params object[] list)
            {
                NodeGrid.Clear();
                NodeContentCollection.NodeContents.Clear();
                CreateGrid();
            }
        }

    }
}
```

Code Backup / Repository

The Starship One project has been developed and saved/backed up using:

- Microsoft OneDrive for local/cloud storage and back up; and
- GitHub local and remote repository: (https://github.com/RobertMair/u3157923_9746_Assessment2_2020) .



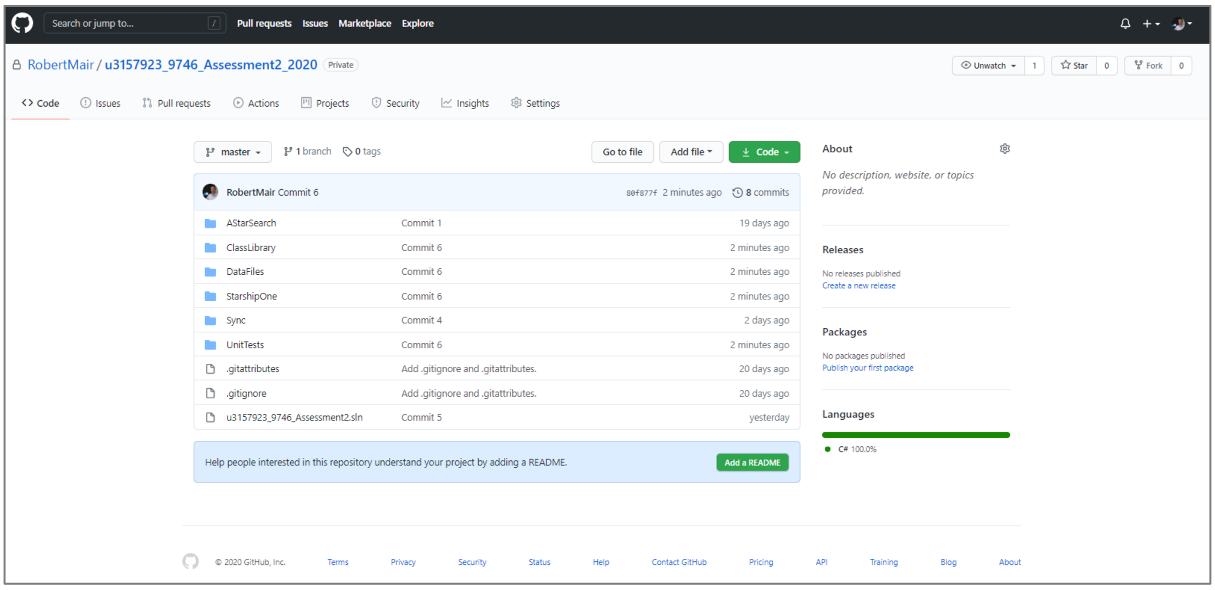
C 0 1

C:u3157923_9746_Assessmen...

C:\...\Robert\OneDrive - azienda.com.au\Degree Web Design\Adv Prog (9746) - 20\u3157923_9746_Assessment2_2020

Name	Size Auto	Modified	Type	Att...
..		14/11/2020 5:16:...	File folder	
ClassLibrary	11.83 MB	15/11/2020 9:54:...	File folder	
StarshipOne	12.34 MB	15/11/2020 9:49:...	File folder	
TestResults	0 Byte(s)	15/11/2020 9:35:...	File folder	A
UnitTests	1.84 MB	15/11/2020 9:34:...	File folder	A
DataFiles	216.27 MB	15/11/2020 8:15:...	File folder	
packages	65.09 MB	14/11/2020 5:08:...	File folder	A
Sync	73 kB	13/11/2020 1:23:...	File folder	A
AStarSearch	101 kB	27/10/2020 12:1...	File folder	A
.vs	3.36 MB	26/10/2020 5:42:...	File folder	
AAA - Readme - Docs and Videos	0 Byte(s)	20/10/2020 10:5...	File folder	
u3157923_9746_Assessment2.sln	3 kB	14/11/2020 5:08:...	Visual Studio Solution	A
.gitignore	6 kB	27/10/2020 9:50:...	Text Document	A
.gitattributes	3 kB	27/10/2020 9:50:...	Text Document	A

Figure 6. Microsoft OneDrive used for local/cloud storage and backup.



Search or jump to... Pull requests Issues Marketplace Explore

RobertMair / u3157923_9746_Assessment2_2020 Private

Unwatch 1 Star 0 Fork 0

Code Issues Pull requests Actions Projects Security Insights Settings

master 1 branch 0 tags Go to file Add file Code

RobertMair Commit 6 80f877f 2 minutes ago 8 commits

AStarSearch	Commit 1	19 days ago
ClassLibrary	Commit 6	2 minutes ago
DataFiles	Commit 6	2 minutes ago
StarshipOne	Commit 6	2 minutes ago
Sync	Commit 4	2 days ago
UnitTests	Commit 6	2 minutes ago
.gitattributes	Add .gitignore and .gitattributes.	20 days ago
.gitignore	Add .gitignore and .gitattributes.	20 days ago
u3157923_9746_Assessment2.sln	Commit 5	yesterday

Help people interested in this repository understand your project by adding a README. Add a README

About

No description, website, or topics provided.

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

C# 100.0%

© 2020 GitHub, Inc. Terms Privacy Security Status Help Contact GitHub Pricing API Training Blog About

Figure 7. GitHub used for local and remote repository.

Testing and Verification

During development of Starship One a lot of manual testing has been conducted. In addition, two unit tests have also been setup in Visual Studio per:

NodeList Unit Test

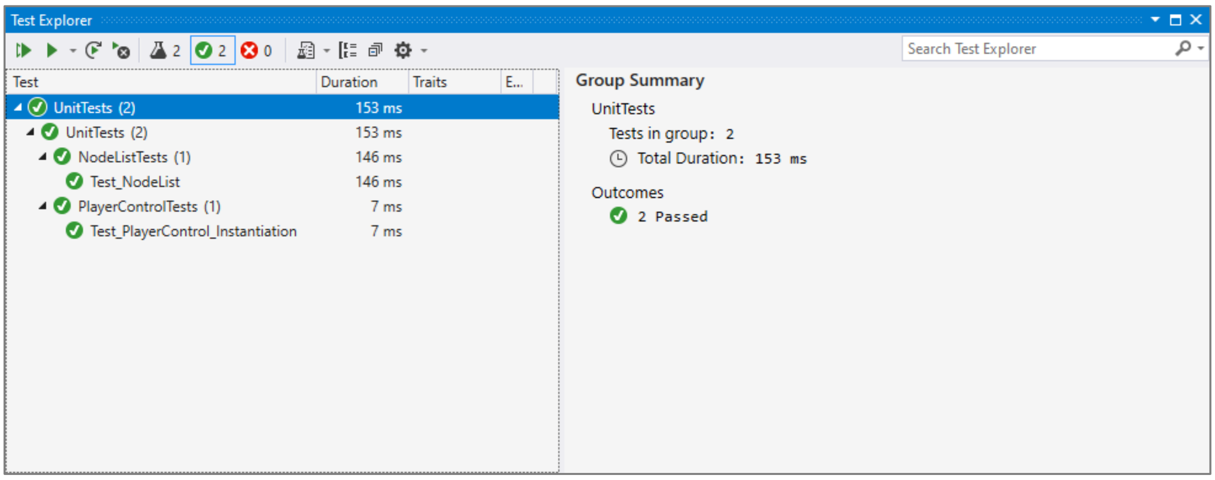
```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using u3157923_9746_Assessment2;

namespace UnitTests
{
    [TestClass]
    public class NodeListTests
    {
        [TestMethod]
        public void Test_NodeList()
        {
            const int expectedElements = 23;
            NodeList.SetNodes();
            Assert.IsNotNull(NodeList.NodeContentTypes);
            Assert.AreEqual(expectedElements, NodeList.NodeContentTypes.Count, $"E
xpected {expectedElements} elements");
        }
    }
}
```

PlayerControl Unit Test

```
using ClassLibrary;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using u3157923_9746_Assessment2;

namespace UnitTests
{
    [TestClass]
    public class PlayerControlTests
    {
        [TestMethod]
        public void Test_PlayerControl_Instantiation()
        {
            var inputRecord = new GuiInput.INPUT_RECORD { EventType = GuiInput.KEY
_EVENT };
            uint recordLen = 1;
            var f = true;
            var result = PlayerControl.PlayerInput(new GuiInput.ConsoleHandle(), r
ef inputRecord, ref recordLen, ref f);
            Assert.IsNotNull(result, "expecting result");
            Assert.AreEqual(0, result, "expecting 0");
        }
    }
}
```

Unit Test Results

This is the report generated in Visual Studio when the unit tests are run.



Test Explorer

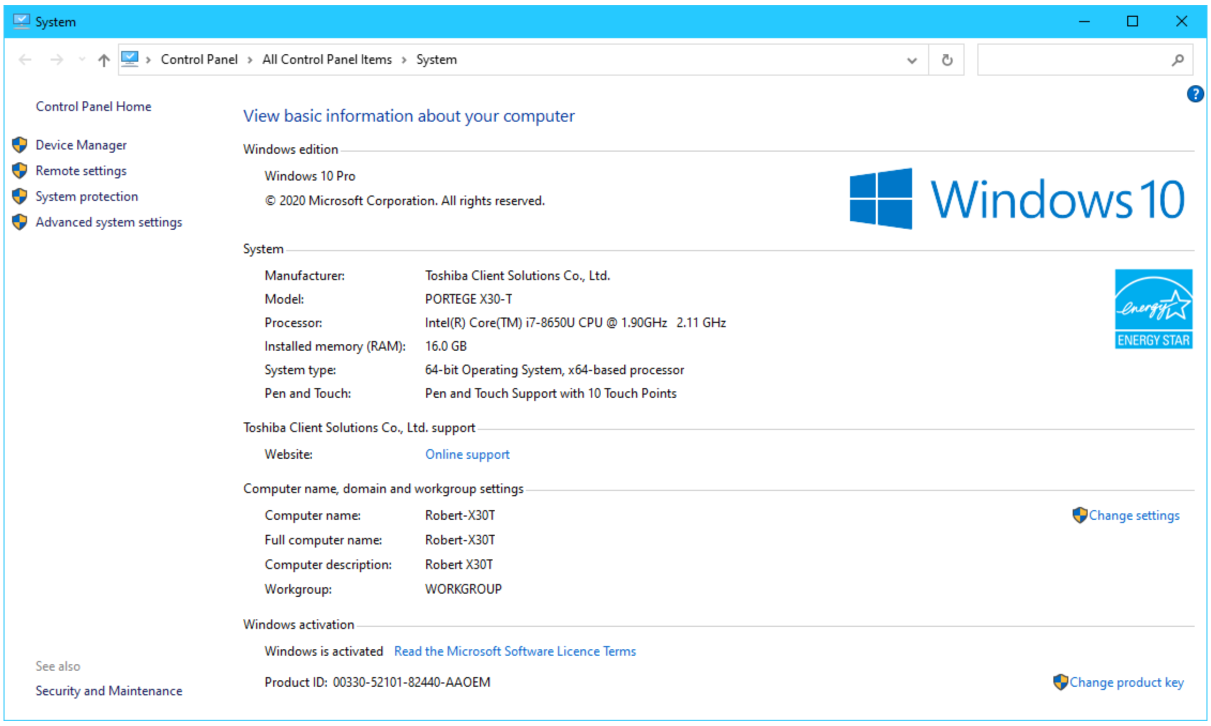
Test	Duration	Traits	E...
UnitTests (2)	153 ms		
UnitTests (2)	153 ms		
NodeListTests (1)	146 ms		
Test_NodeList	146 ms		
PlayerControlTests (1)	7 ms		
Test_PlayerControl_Instantiation	7 ms		

```
Group Summary
  UnitTests
    Tests in group: 2
    Total Duration: 153 ms

  Outcomes
    2 Passed
```

Figure 8. Visual Studio unit test results.

Test Plan

Windows device(s) tested:



System

Control Panel > All Control Panel Items > System

View basic information about your computer

Windows edition

Windows 10 Pro
© 2020 Microsoft Corporation. All rights reserved.

System

Manufacturer:	Toshiba Client Solutions Co., Ltd.
Model:	PORTEGE X30-T
Processor:	Intel(R) Core(TM) i7-8650U CPU @ 1.90GHz 2.11 GHz
Installed memory (RAM):	16.0 GB
System type:	64-bit Operating System, x64-based processor
Pen and Touch:	Pen and Touch Support with 10 Touch Points

Toshiba Client Solutions Co., Ltd. support

Website: Online support

Computer name, domain and workgroup settings

Computer name:	Robert-X30T
Full computer name:	Robert-X30T
Computer description:	Robert X30T
Workgroup:	WORKGROUP

Windows activation

Windows is activated Read the Microsoft Software Licence Terms

Product ID: 00330-52101-82440-AAOEM

Figure 9: Toshiba laptop with Windows 10 used in testing

List of functionalities tested and results

Functionality	Result
Running the game	
Building the game in Visual Studio 2019	Passed

Game operates in ‘console’	Passed
User input for player character movement and menu items works	Passed
Splash screen exists and displays correctly after starting the game	Passed
User Interface	
Check all graphic elements, texts and animations display in the console	Passed
Check the ability to choose any start level from the start menu	Passed
Check the ability to pause on any game stage	Passed
Check the ability to turn the in-game music on/off	Passed
Check the ability to turn the in-game sound FX on/off	Passed
Gameplay	
Check player character can collect ‘collectibles’ and this updates inventory	Passed
Check player character can place ‘defensive’ items in the game area	Passed
Check NPC movement	Passed
Check NPC/Player character ‘collision detection’ Screen update sync is not 100% aligned for player movement and NPC movement which can result in them ‘passing through’ one another. This is to do with the ingame timer that requires more development to ensure all player input, NPC movement and screen updates are synced correctly in the game loop.	Warning

Figure 10: Table of test functionalities

Bibliography

[i] Wikipedia. 2020. Andromeda Galaxy - Wikipedia. [ONLINE] Available at: https://en.wikipedia.org/wiki/Andromeda_Galaxy. [Accessed 02 August 2020].

[ii] Wikipedia. 2020. Electro-technical officer - Wikipedia. [ONLINE] Available at: https://en.wikipedia.org/wiki/Electro-technical_officer. [Accessed 02 August 2020].

[iii] Google Docs. 2020. One Page Game Design Document Template - Google Docs. [ONLINE] Available at: https://docs.google.com/document/d/1npEvqcMZSp0IX2hWw6Qq0WqJVfmVqS_YOGFWnnwfh-A/edit#heading=h.1s1dfwgzecqh. [Accessed 09 August 2020].

[iv] Wikipedia. 2020. Roguelike - Wikipedia. [ONLINE] Available at: https://en.wikipedia.org/wiki/Roguelike. [Accessed 09 August 2020].

[v] Wikipedia. 2020. Amstrad CPC - Wikipedia. [ONLINE] Available at: https://en.wikipedia.org/wiki/Amstrad_CPC. [Accessed 02 August 2020].

[vi] Wikipedia. 2020. Commodore 64 - Wikipedia. [ONLINE] Available at: https://en.wikipedia.org/wiki/Commodore_64. [Accessed 02 August 2020].

[vii] Wikipedia. 2020. ZX Spectrum - Wikipedia. [ONLINE] Available at: https://en.wikipedia.org/wiki/ZX_Spectrum. [Accessed 02 August 2020].

[viii] ASCII - RogueBasin. 2020. ASCII - RogueBasin. [ONLINE] Available at: http://www.roguebasin.com/index.php?title=ASCII. [Accessed 09 August 2020].

[ix] Video modes - CPCWiki. 2020. Video modes - CPCWiki. [ONLINE] Available at: http://www.cpcwiki.eu/index.php/Video_modes#Colour_attributes. [Accessed 09 August 2020].

[x] Google Image Search using “Ryan Wolfe of Ki Ryn Studios”. [ONLINE] Available at: https://bit.ly/3a9eE1s. [Accessed 09 August 2020].

[xi] Wikipedia. 2020. La Abadía del Crimen - Wikipedia. [ONLINE] Available at: https://en.wikipedia.org/wiki/La_Abadía_del_Crimen. [Accessed 15 August 2020].

[xii] The Music System - CPCWiki. 2020. The Music System - CPCWiki. [ONLINE] Available at: http://www.cpcwiki.eu/index.php/The_Music_System. [Accessed 09 August 2020].

[xiii] Skiff’s, E. (2020). Free 8-bit Music - Eric Skiff’s Resistor Anthems. [online] Ericskiff.com. Available at: https://ericskiff.com/music/ [Accessed 14 Nov. 2020].

[xiv] SubspaceAudio (2016). 512 Sound Effects (8-bit style). [online] OpenGameArt.org. Available at: https://opengameart.org/content/512-sound-effects-8-bit-style [Accessed 14 Nov. 2020].

[xv] Wikipedia. 2020. Rogue (video game) - Wikipedia. [ONLINE] Available at: https://en.wikipedia.org/wiki/Rogue_(video_game). [Accessed 08 August 2020].

[xvi] Home of Angband - rephial.org. 2020. Home of Angband - rephial.org. [ONLINE] Available at: http://rephial.org. [Accessed 08 August 2020].

[xvii] NetHack 3.6.6: NetHack Home Page. 2020. NetHack 3.6.6: NetHack Home Page. [ONLINE] Available at: https://www.nethack.org/index.html. [Accessed 08 August 2020].

[xviii] New Game: Lab Escape. 2020. New Game: Lab Escape. [ONLINE] Available at: http://www.cpcwiki.eu/forum/games/new-game-lab-escape/. [Accessed 02 August 2020].

[xix] CPC Game Reviews - P. 2020. CPC Game Reviews - P. [ONLINE] Available at: http://www.cpcgamereviews.com/p/index.html. [Accessed 02 August 2020].

[xx] Indie Retro News: Rogue +6DGM - A nice C64 roguelike fixed and fully playable by Hokuto Force. [ONLINE] Available at: http://www.indieretronews.com/2017/06/rogue-6dgm-nice-c64-roguelike-fixed-and.html. [Accessed 02 August 2020].

[xxi] Ranarama - C64-Wiki. 2020. Ranarama - C64-Wiki. [ONLINE] Available at: https://www.c64-wiki.com/wiki/Ranarama. [Accessed 02 August 2020].

[xxii] CPC Game Reviews - W. 2020. CPC Game Reviews - W. [ONLINE] Available at: http://www.cpcgamereviews.com/w/index.html. [Accessed 02 August 2020].

[xxiii] Spectrum Computing. 2020. Mystic Tower at Spectrum Computing - Sinclair ZX Spectrum games, software and hardware. [ONLINE] Available at: https://spectrumcomputing.co.uk/index.php?cat=96&id=3351. [Accessed 02 August 2020].

[xxiv] Indie Retro News: Escape from Cnossus - A roguelike game for the ZX Spectrum. 2020. Indie Retro News: Escape from Cnossus - A roguelike game for the ZX Spectrum. [ONLINE] Available at: http://www.indieretronews.com/2015/02/escape-from-cnossus-roguelike-game-for.html. [Accessed 02 August 2020].

[xxv] World of Spectrum. 2020. New game: Magical Tower Adventure - Mini version - World of Spectrum. [ONLINE] Available at: https://worldofspectrum.org/forums/discussion/52699/new-game-magical-tower-adventure-mini-version. [Accessed 02 August 2020].

[xxvi] Wikipedia. 2020. Time Bandit - Wikipedia. [ONLINE] Available at: https://en.wikipedia.org/wiki/Time_Bandit. [Accessed 08 August 2020].

[xxvii] Wikipedia Contributors (2020). User story. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/User_story [Accessed 19 Oct. 2020].

[xxviii] The Three Rs or the Connextra format (2020). User Experience Mapping. [online] O'Reilly Online Learning. Available at: https://www.oreilly.com/library/view/user-experience-mapping/9781787123502/92d21fe3-a741-49ff-8200-25abf18c98d0.xhtml [Accessed 19 Oct. 2020].

[xxix] Wikipedia Contributors (2020). INVEST (mnemonic). [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/INVEST_(mnemonic) [Accessed 19 Oct. 2020].

[xxx] Wikipedia Contributors (2020). Entity–relationship model. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model [Accessed 15 Nov. 2020].