The second phase of my IT PAT for 2018.

IT PAT Phase 2

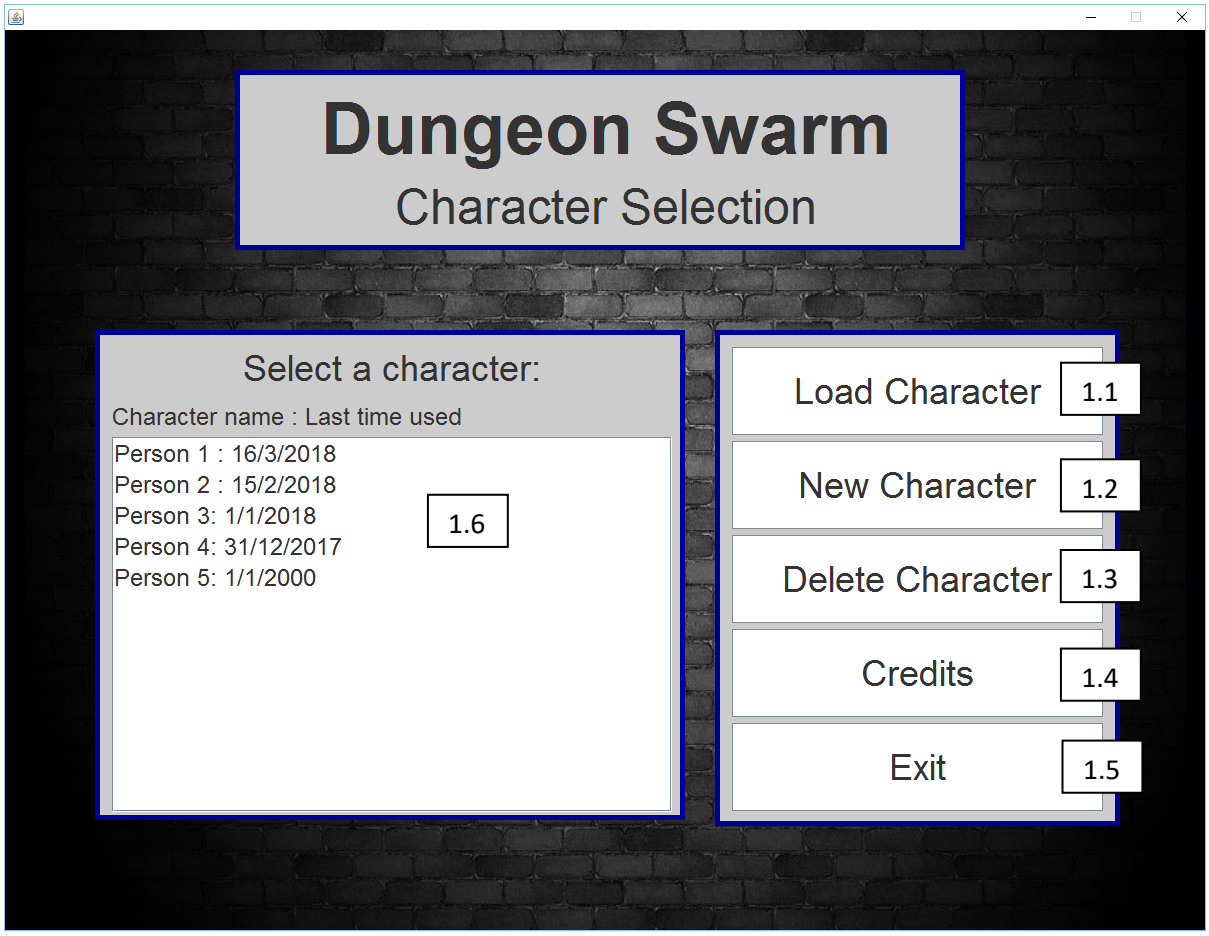
2018

Ralph McDougall

Contents

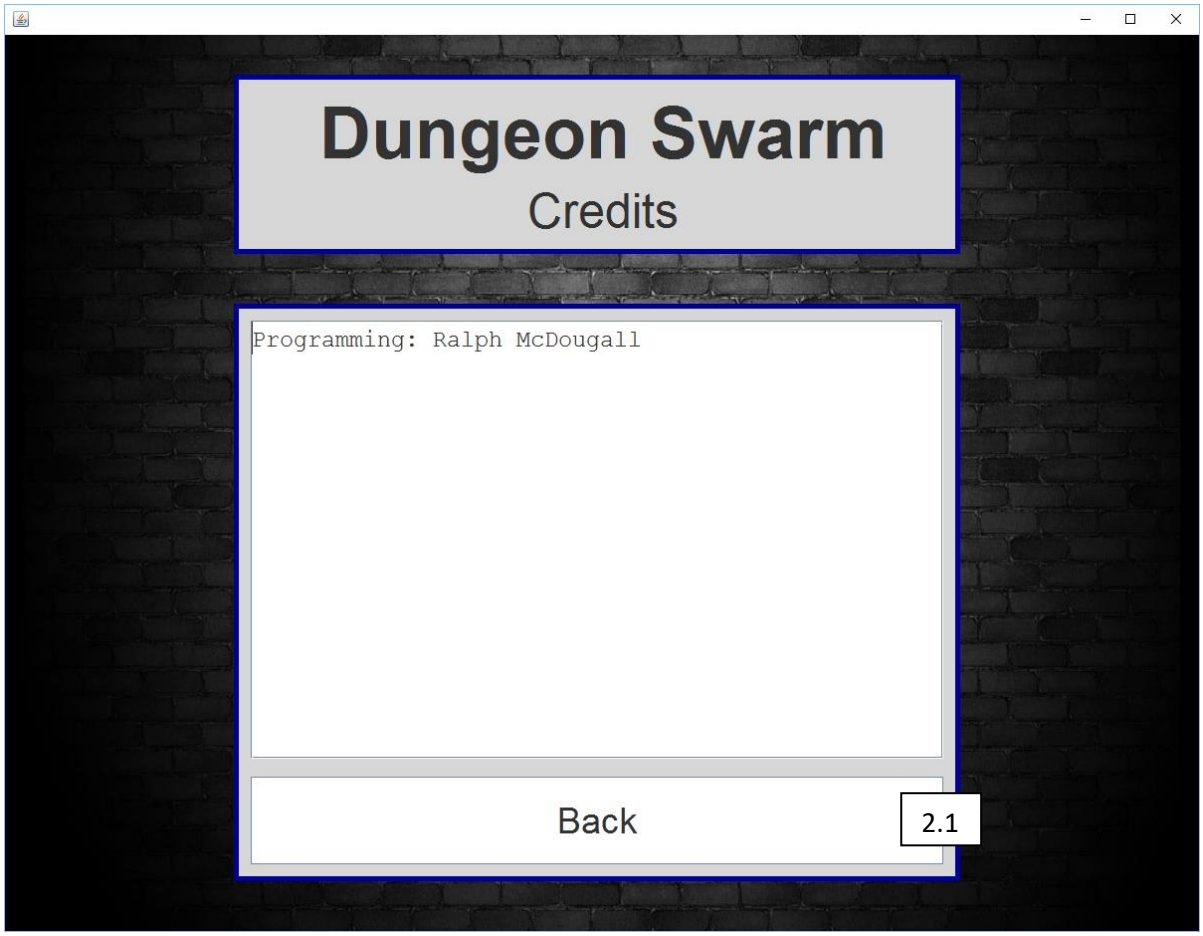
User Interface Design

Screen 1 (Character selection screen):



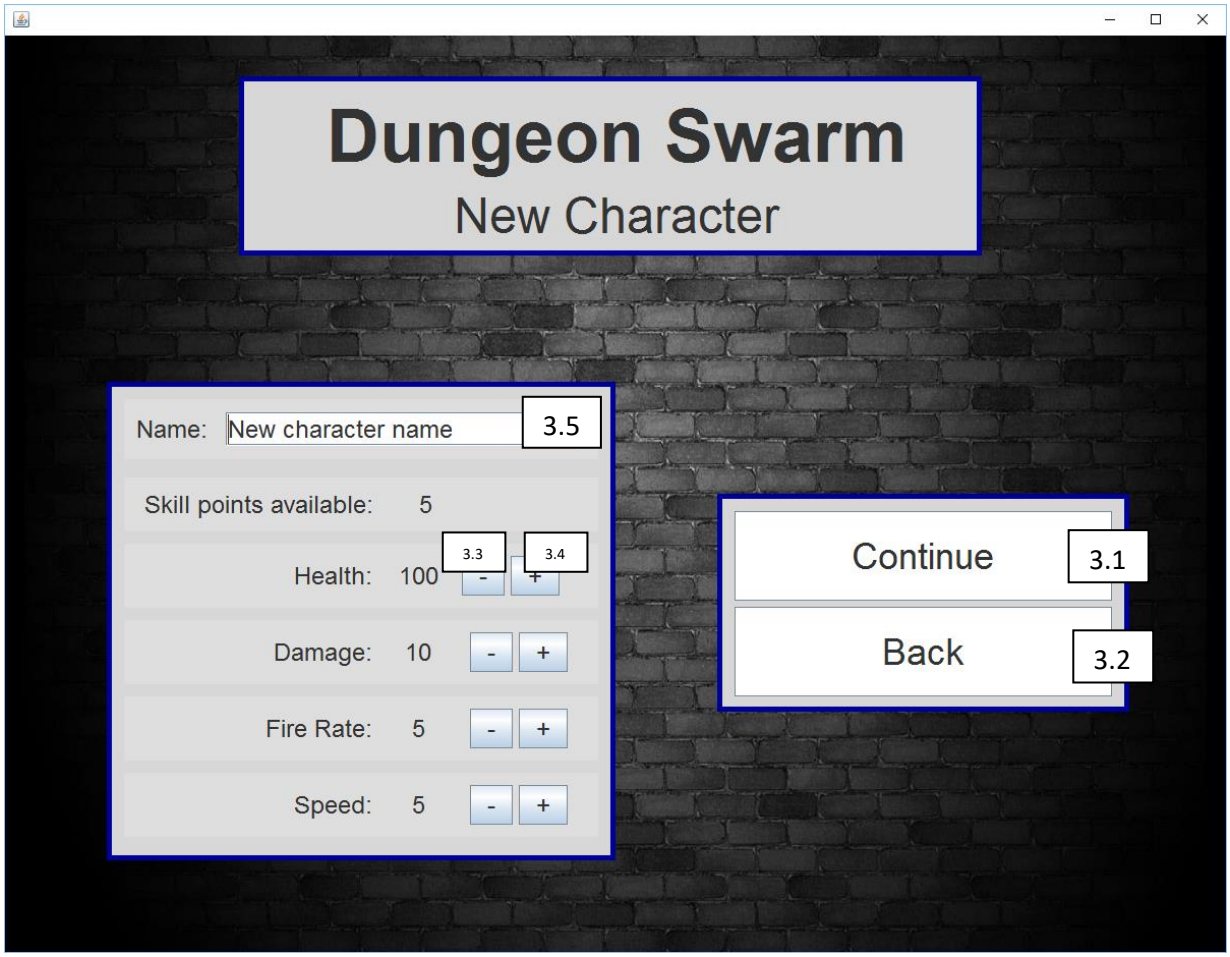
Action Element	Description
1.1	A button that will load the selected character and then open *Screen 4* and close *Screen 1*. If no character is selected in the list, a message box will appear informing the user that they need to select a character or create a new one.
1.2	A button that opens *Screen 3* and closes *Screen 1*.
1.3	A button that removes the selected character. A message box will open informing the user to select a character if they have not yet selected one. A confirmation box asks the user if they are sure they want to delete that character.
1.4	A button that opens *Screen 2* and closes *Screen 1*.
1.5	A button that closes the application and frees all resources and makes sure all changes to persistent data has been recorded.
1.6	A list of all characters created in the past. It also indicates the last time that character was used in the game. The entries are sorted so that the most recently used character is on top.

Screen 2 (Credits screen):



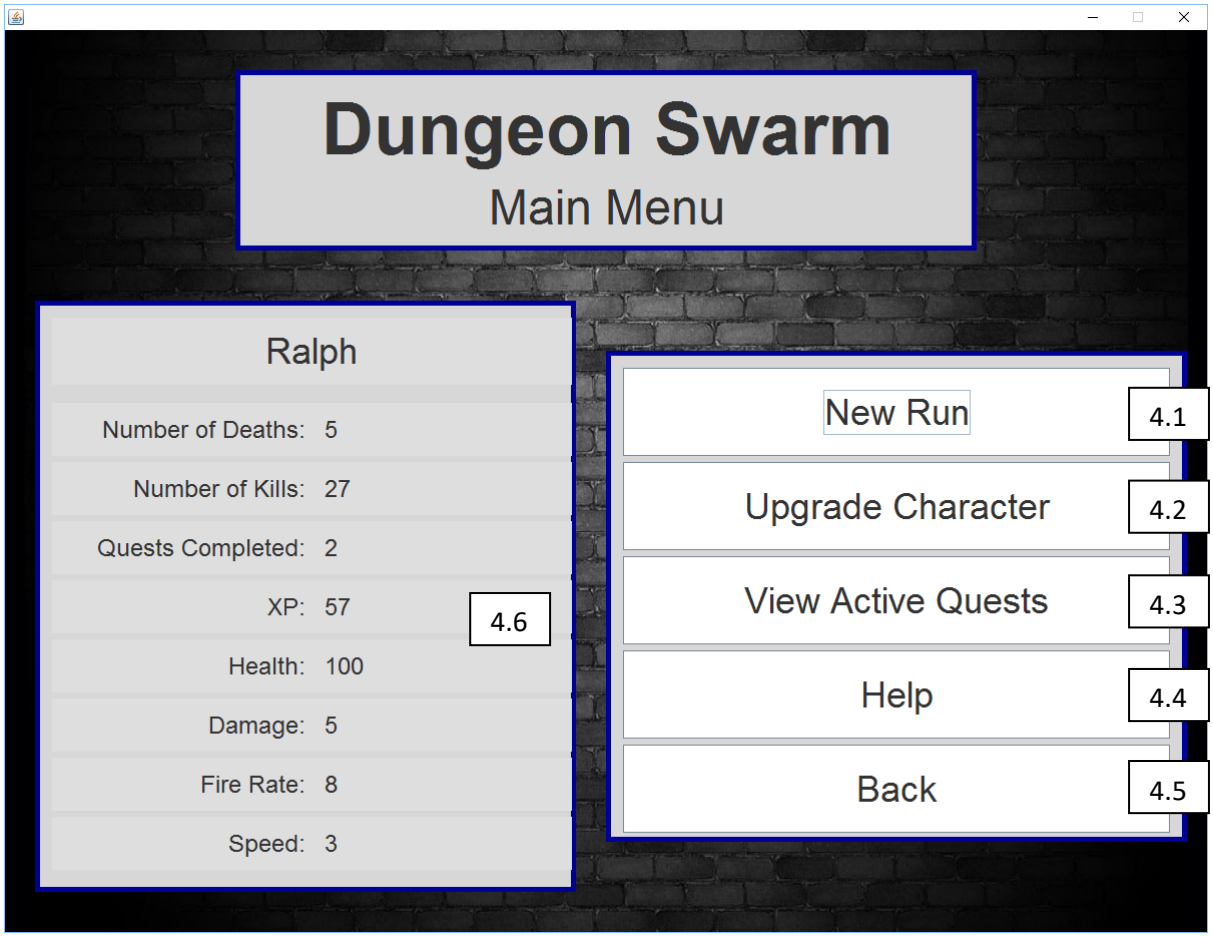
Action Element	Description
2.1	A button that re-opens *Screen 1* and closes *Screen 2*.

Screen 3 (New character screen):



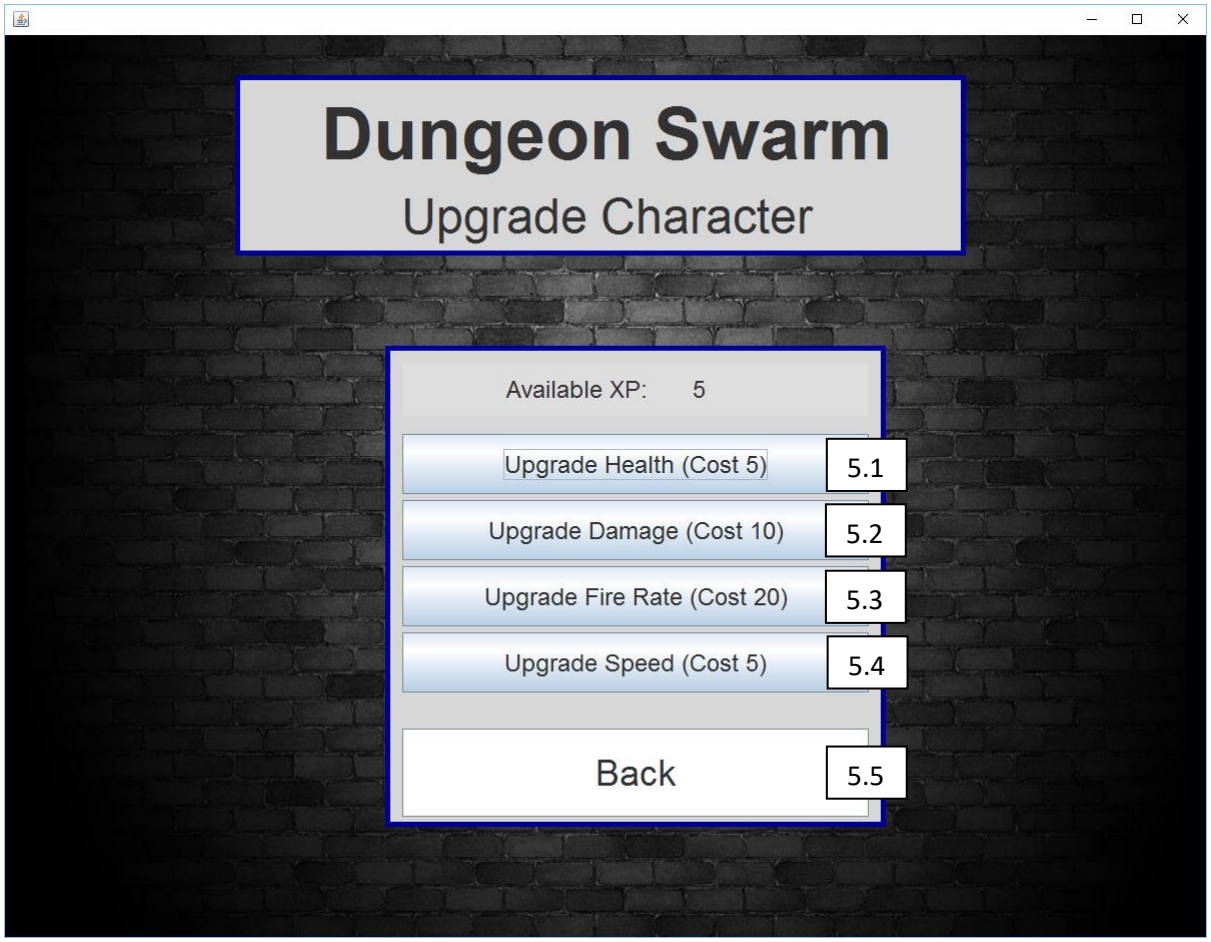
Action Element	Description
3.1	A button that opens *Screen 4* and creates a new character with the chosen name and skills. If the name of the character is invalid, a message box asks the user to choose a different name.
3.2	A button that re-opens *Screen 1* and closes *Screen 3*.
3.3	Each button in this column decreases the displayed value next to it and increases the number of skill points available.
3.4	Each button in this column increases the displayed value next to it and decreases the number of skill points available. If no skill points are available, no change will be noted.
3.5	A text field where the user can enter the name of the new character.

Screen 4 (Character main menu screen):



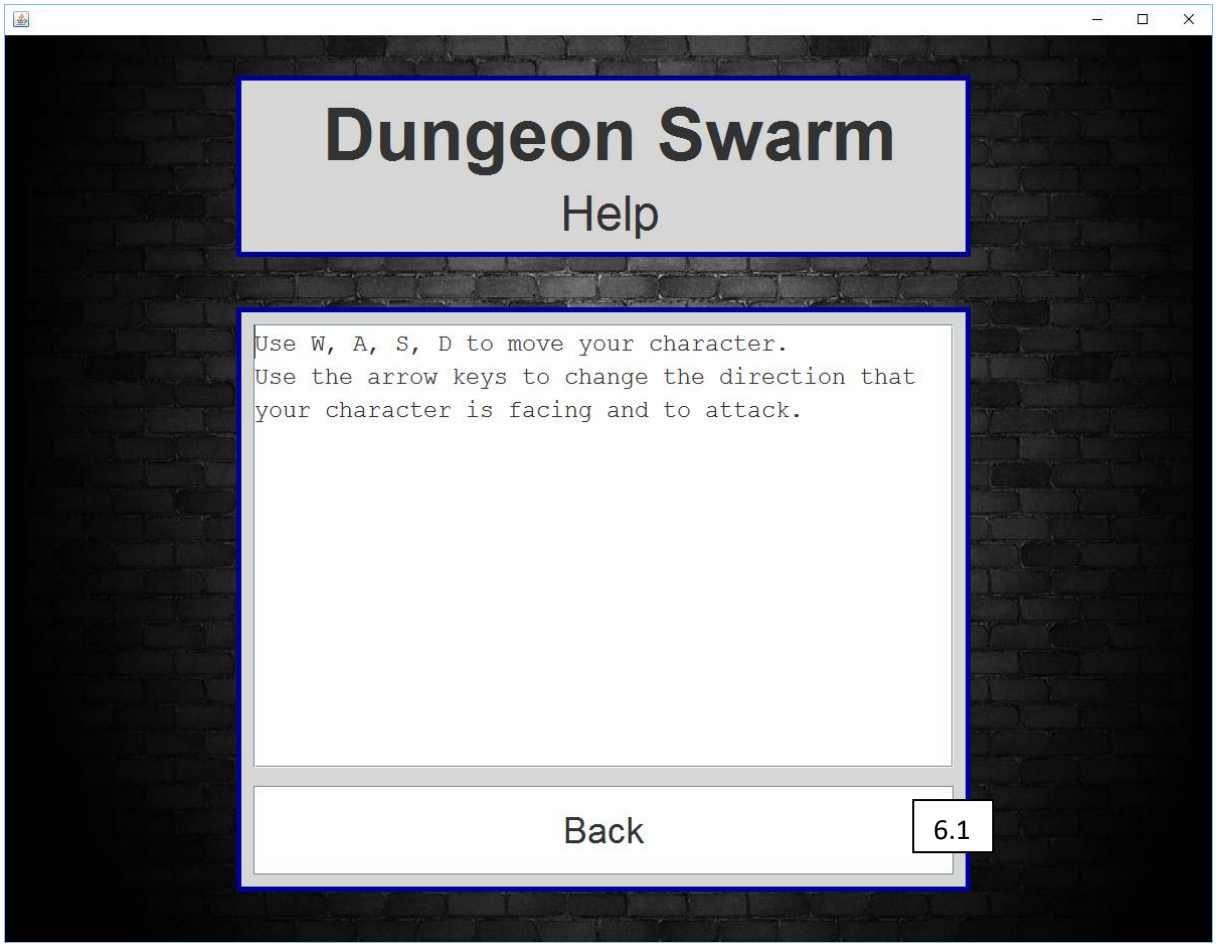
Action Element	Description
4.1	A button that opens *Screen 8* and closes *Screen 4*. This starts a new game.
4.2	A button that opens *Screen 5* and closes *Screen 8*.
4.3	A button that opens *Screen 7* and closes *Screen 8*.
4.4	A button that opens *Screen 6* and closes *Screen 8*.
4.5	A button that re-opens *Screen 1* and closes *Screen 4*.
4.6	All of these labels are dynamic and display the player's stats.

Screen 5 (Upgrade character screen):



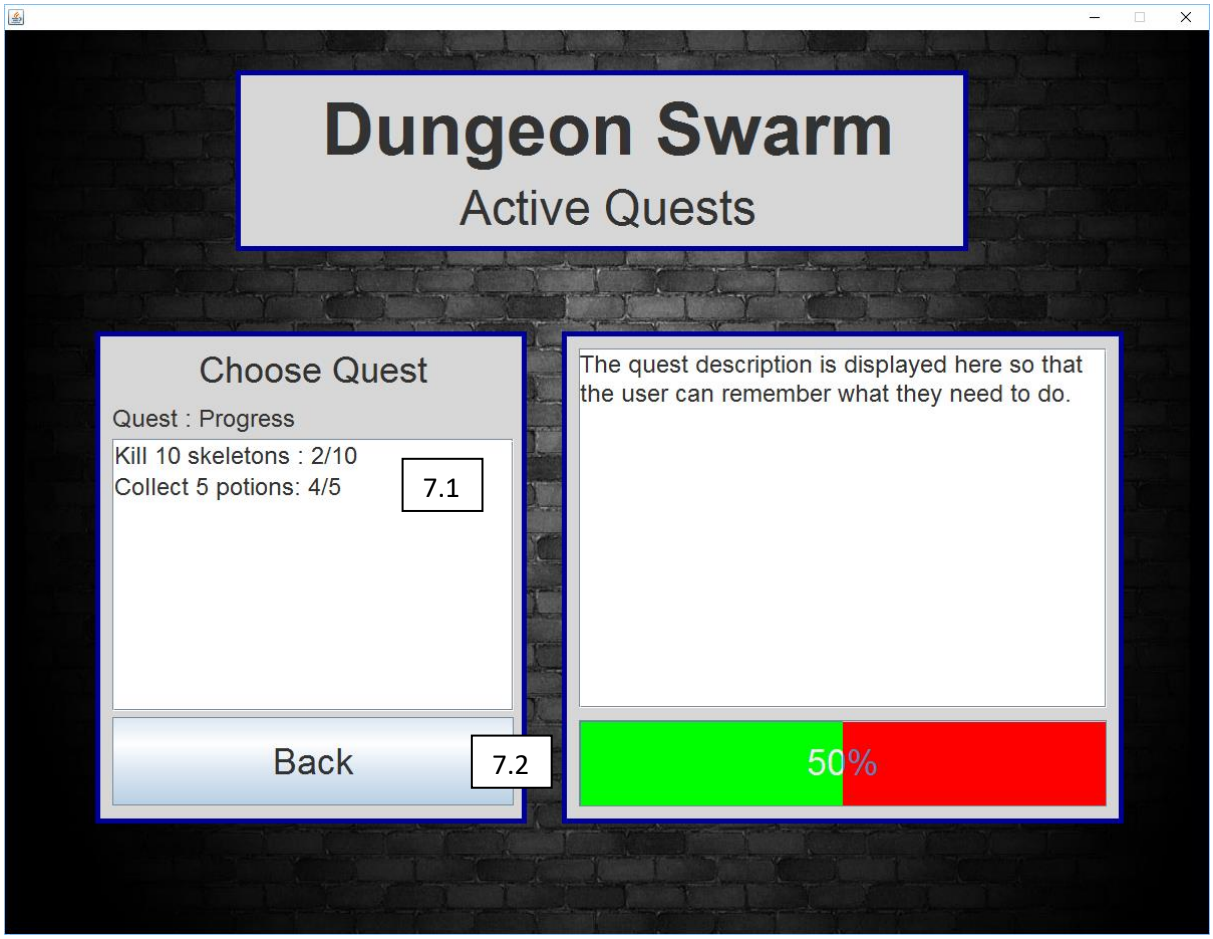
Action Element	Description
5.1	A button that increases the “Health” stat and reduces the available XP.
5.2	A button that increases the “Damage” stat and reduces the available XP.
5.3	A button that increases the “Fire Rate” stat and reduces the available XP.
5.4	A button that increases the “Speed” stat and reduces the available XP.
5.5	A button that re-opens *Screen 4* and closes *Screen 5*.

Screen 6 (Help screen):



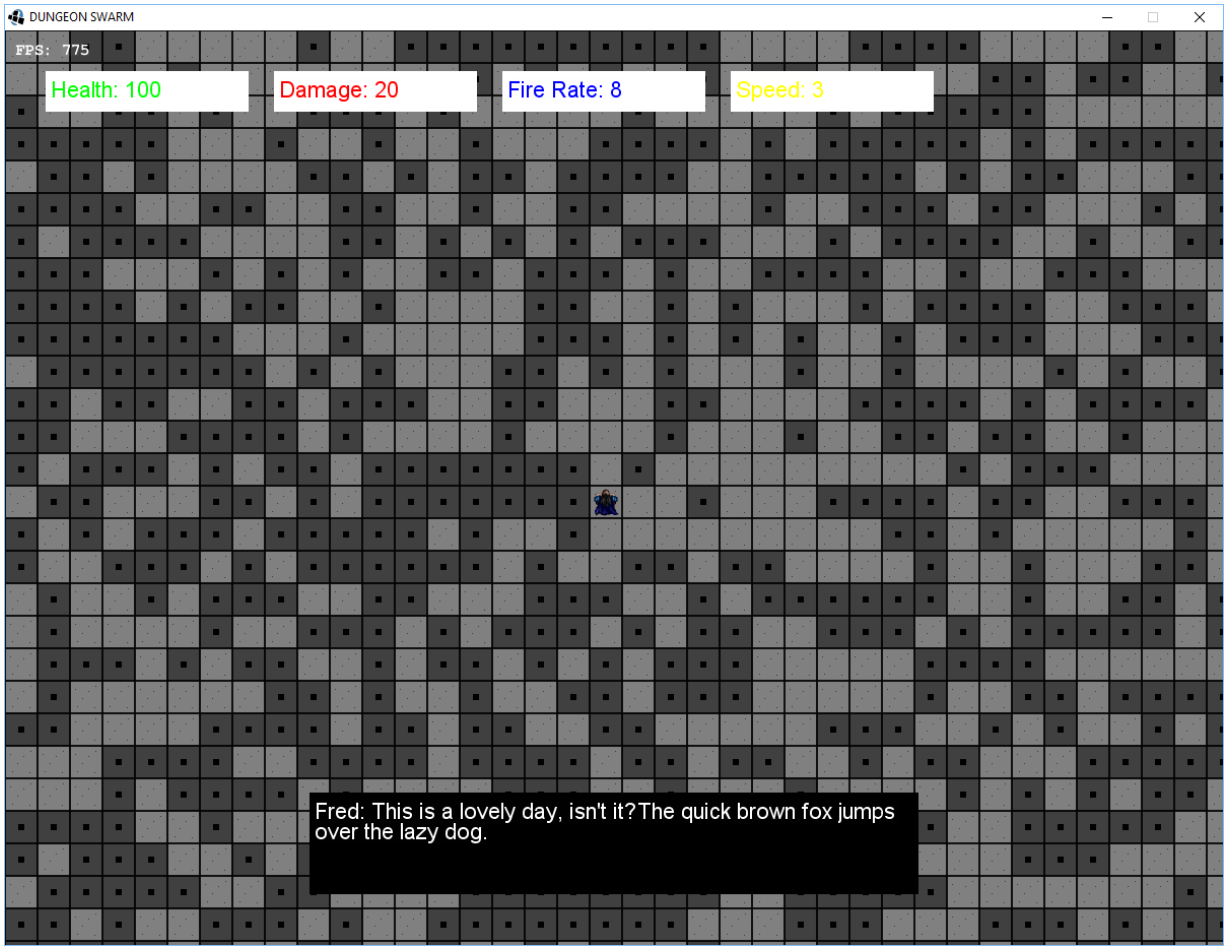
Action Element	Description
6.1	A button that re-opens *Screen 4* and closes *Screen 6*.

Screen 7 (Active quests screen):



Action Element	Description
7.1	A list that indicates the current active quests as well as the progress of each quest.
7.2	A button that re-opens *Screen 4* and closes *Screen 7*.

Screen 8 (Main game screen):



Action Element	Description
ESC	Closes *Screen 8* and re-opens *Screen 4*. First asks the user if they are sure they want to exit the current run.
W, A, S, D	Moves the character on the screen.
Arrows	Changes direction character is facing and shoots a projectile in that direction.
E	Allows the character to interact with non-player characters and pick-up items.
Message box	The message box displays what the NPCs are saying and shows a description for items that are picked up.
Stat displays	The stats change as the player picks up items or as they take damage.

Sequencing

Event	Program flow
Game system	
Program start	• Load the game logger. The logger allows for easier debugging as the exact conditions that caused a bug can be reviewed. • Load the database bridge. • Load the types of tiles from the database so that it doesn't have to be done when the main game starts. • Load the map tile segments containing pre-created blocks of tiles to construct the game map from. • Load the quest formats. • Initialise the screens and make them invisible. • Make *Screen 1* visible.
Program end	• Free all resources. • Close screens. • System exit.
Screen 1	
***Screen 1* made visible**	• Retrieve previous characters from database. • Update list of characters. • Display screen.
***"Load Character"* button pressed**	• Retrieve selected character. • If the character isn't valid, ask for a valid character. • Retrieve the stats of the chosen character and set those to the current player stats. • Make *Screen 4* visible.
***"New Character"* button pressed**	• Make *Screen 3* visible.
Screen 3	
***"Continue"* button pressed**	• Check if the name entered is valid, if not, ask the user for a different name. • Retrieve all stats for the new player and set those to be the current player statistics. • Make *Screen 4* visible.
"-" button pressed	• Check if the stat is higher than the minimum possible, if not, inform the user that this stat cannot be lowered. • Lower the stat and add a skill point.
***"+"* button pressed**	• Increase the stat and remove a skill point.
Screen 4	
***Screen* 4 made visible**	• Set character name being displayed to the name of the current character. • Set the displayed stats to the stats of the selected character.
Screen 5	
***Screen 5* made visible**	• Set available XP and the requirement for upgrade to the current player's amounts.
***"Upgrade"* buttons pressed**	• Reduce available XP. • Increase selected stat. • Update button display to new value.

Screen 7	
***Screen 7* made visible**	• Fill the list with the current player's active quests arranged in alphabetical order.
List item selected	• Update progress bar and description with selected quest description and progress.
Screen 8	
***Screen 8* made visible**	• Load sprites. • Create new map. If sufficiently many levels have been completed in this run, load a boss level instead of a normal level. • Place player at map starting point. • Set text displayed on the heads-up-display to match the character's stats.
Create map	• Pick new map width and height. • Pick random starting cell in map grid. • Perform randomised breadth first search from starting cell until sufficiently many cells have been reached. • Retrieve all possible map segments. • Fill selected cells with map segments that fit cell alignment with neighbouring chosen cells. • Select random endpoint as the maze exit. • Select random cell to place item. • Select random cell to place random NPC. • Assign either a quest or dialogue to the NPC. • If a quest is to be issued, generate a new quest.
Update game	• Check if the player character still has enough health to continue, if not, take them back to *Screen 4*. • Move player character if the user uses the W, A, S, D keys. • Change the direction that the character faces and shoot a projectile if the user uses the arrow keys. • If the player is close enough to an NPC or an item and the user uses the E key, interact with it. Display a message box with the item description of NPC dialogue. • Spawn new enemy if necessary. • Perform path-finding to allow enemies to track player. • Move enemies and allow enemies to shoot projectiles at player. • Move all projectiles and check if they have hit an enemy or the player. • Check if any quests have progressed. • Check if the player has reached the end of the maze, if they have, create a new map.
Render to screen	• Display all game tiles. • Display all characters and items and projectiles. • Display the heads-up-display.
Spawn new enemy	• Spawn random enemy somewhere on map out of the user's frame of view.
Path-find	• Start at the player's location and breadth first search out a given distance and keep track of a vector at each location for the shortest distance to the player.
Enemy behaviour	• Check how far away the player is, if the player is close enough, shoot a projectile at them, otherwise try to move close to the player with the path-finding.

Shoot projectile	• Spawn projectile at the shooters location with a given direction.
Move entity	• Update entity sprite for the next scene in their animation. • Change the coordinate of the entity to reflect the movement that has occurred.
Create new quest	• Choose name of issuer. • Choose type of quest. • Choose quest target. • Choose quest target count. • Choose quest reward. • Construct display sentence for new quest.
Check for quest progresses	• Every time an item is picked up, an NPC is spoken to or an enemy is killed, check if it matches any of the active quests. • If a quest has been completed, reward the player and remove the quest. • Inform the player that they have completed a quest on the in-game message/dialogue box.
Item pick-up	• Display a message informing the player as to what the item is called. • Update the player's stats.

Class Design

Name	Description
Entity	The class that encompasses all characters, items, projectiles and tiles
worldX	The x-coordinate of the entity on the world map
worldY	The y-coordinate of the entity on the world map
allSprites	The sprites that the entity can use
currentSprite	The sprite that the entity is currently using
direction	The direction that the entity is facing (this affects the sprite)
draw()	Draw the entity sprite onto the screen
getScreenX()	Get the x-position that the entity should be drawn on the screen.
getScreenY()	Get the y-position that the entity should be drawn on the screen.
moveWorldX()	Move the entity in the x-direction
moveWorldY()	Move the entity in the y-direction

Character (extends Entity)	The class that encompasses all enemies, NPCs and the player.
health	The character's health.
damage	The damage that the character deals.
fireRate	The rate at which the character can fire.
speed	The speed at which the character can move.
projectileType	The type of projectile the character shoots.
lastTimeFired	The last time the character shot a projectile.
faction	Whether the character is hostile or friendly.
name	The name of the character.
fireProjectile()	Shoot a new projectile.
checkAlive()	Check that the character shouldn't be removed from the game because they have run out of health.
remove()	Remove the character from the game.

Enemy (extends Character)	The class that controls the behaviour of enemies.
trackPlayer()	Move the enemy towards the player character.
update()	Check if it is still alive and move it if necessary.

Projectile (extends Entity)	The class the describes all projectiles fired by characters.
damage	The damage that the projectile needs to inflict.
faction	The faction of the shooter.
speed	The speed at which the projectile moves.
distanceTravelled	The distance that the projectile has travelled
maxDist	The maximum distance that the
update()	Move the projectile and make sure it hasn't collided with anything or travelled too far.

Map	
tiles	A grid of all of the tiles on the map.
width	The width of the map.
height	The height of the map.
generateMap()	Generates a new random map and initialises all of the tiles.
generateTiles()	Fills the grid with tiles.
Tile (extends Entity)	This keeps track of all of the properties of each tile.
walkable	Whether or not characters can walk on this tile.
tileID	The ID of the tile type, this allows for easily checking what each tile is.
Item (extends Entity)	This handles all items that need to be placed on the grid.
name	The name of the item.
description	Flavour text description of the item.
healthAffect	The affect that this item has on a player's health.
damageAffect	The affect that this item has on a player's damage.
fireRateAffect	The affect that this item has on a player's fire rate.
speedAffect	The affect that this item has on a player's speed.
pickUp()	What happens when the item gets picked up by a player.
NPC (extends Character)	This handles everything to do with the Non-Player Characters (friendly characters) including interactions with the player and creating quests.
text	The text that the NPC will display when talking to the player.
issueQuest()	Issue a new quest to the player.
composeQuest()	Create a new quest when this NPC gets initialised.
PlayerCharacter (extends Character)	This makes it easier to distinguish between the player character and all other characters.
checkKeyboardInput()	Check what keys the user has pressed and react accordingly.
pickUpItem()	Attempt to pick up the item that the player is standing next to.
interactWithNPC()	Attempt to talk to the NPC that the player is standing next to.
Particle (extends Entity)	
colour	The colour that the particle displays.
update()	Move the particle some random amount and check if it needs to decay.
Camera	This controls what is visible on the screen at any time and what offset it needs.
width	The number of tiles to show in each row on the screen.

height	The number of tiles to show in each column on the screen.
worldX	The x-coordinate of the top-left block on the screen.
worldY	The y-coordinate of the top-left block on the screen.
moveX()	Move the camera screen in the x-direction.
moveY()	Move the camera screen in the y-direction.

HUDElement	Each element that needs to be displayed on the screen with a label or description.
screenX	The x-coordinate of the top-left of the element on the screen.
screenY	The y-coordinate of the top-left of the element on the screen.
width	The width of the element on the screen.
height	The height of the element on the screen.
text	The text that needs to be displayed.
bgColour	The colour of the background box.
fontColour	The colour of the text that needs to be displayed.
visible	Whether or not the element should be displayed at this time.
draw()	Draw the element on the screen at the appropriate position.
changeText()	Change the text that is being displayed.
setVisible()	Set whether or not the element should be visible on the screen.

Persistent Storage Design

Table 1 (Active Quest Table)

Field Name	Data Type	Description (Optional)
QuestID	AutoNumber	The ID of the current quest
UserID	Number	The ID of the user that it applies to in tblUsers (Foreign key tblUsers)
IssuerName	Short Text	The name of the NPC that issued the quest
QuestType	Number	The ID of the quest type
QuestTargetDescription	Short Text	The Description of the target of the quest (skeleton, potion, etc.)
DesiredTargetCount	Number	The number of times an event needs to be performed to complete the quest
QuestProgress	Number	The number of times the event has already been performed

Table 2 (Enemy Type Table)

Field Name	Data Type	Description (Optional)
EnemyID	AutoNumber	The ID of the enemy type
EnemyName	Short Text	The name of the enemy type
SpriteSheetX	Number	The X coordinate of the first sprite on the sprite sheet
SpriteSheetY	Number	The Y coordinate of the first sprite on the sprite sheet
EnemyHealth	Number	The starting health of the enemy
EnemyMovementSpeed	Number	The movement speed of the enemy
EnemyDamage	Number	The damage that the enemy deals
EnemyDamageType	Number	The type of projectile that the enemy fires (Foreign key tblProjectileTypes)
EnemyDamageRate	Number	The rate at which the enemy deals damage

Table 3 (Items Table)

Field Name	Data Type	Description (Optional)
ItemID	AutoNumber	The ID of the item
ItemName	Short Text	The name of the item
ItemDescription	Short Text	The description text that informs the user of the item's properties
Rarity	Number	How rare it is to find the item (The higher the number, the rarer it is)
SpriteSheetX	Number	The x-coordinate of the item's sprite on the item sprite sheet
SpriteSheetY	Number	The y-coordinate of the item's sprite on the item sprite sheet
HealthBoost	Number	The effect that the item has on the player's health
DamageBoost	Number	The effect that the item has on the player's damage
FireRateBoost	Number	The effect that the item has on the player's fire rate
SpeedBoost	Number	The effect that the item has on the player's speed

Table 4 (Projectile Type Table)

Field Name	Data Type	Description (Optional)
ProjectileTypeID	AutoNumber	The ID of the projectile type
ProjectileTypeName	Short Text	The name of the projectile type
SpriteSheetX	Number	The x-coordinate of the sprite on the sprite sheet
SpriteSheetY	Number	The y-coordinate of the sprite on the sprite sheet
Range	Number	The maximum range that the projectile can travel

Table 5 (Quest Targets Table)

Field Name	Data Type	Description (Optional)
QuestTypeID	Number	The ID of the quest type that this matches with
TargetDescription	Short Text	The description of the target of the quest
MaxTargetCount	Number	The maximum target count for a quest with this target

Table 6 (Quest Type Table)

Field Name	Data Type	Description (Optional)
QuestTypeID	AutoNumber	The ID of the quest type
QuestTypeDescriptor	Short Text	The descriptor for this quest type that is to be used when describing the quest

Table 7 (Tile Type Table)

Field Name	Data Type	Description (Optional)
TileID	AutoNumber	The ID of the tile
TileName	Short Text	The name of the tile
SpriteSheetX	Number	The X coordinate of the tile on the sprite sheet
SpriteSheetY	Number	The Y coordinate of the tile on the sprite sheet
Walkable	Yes/No	Whether or not the player can walk on the tile

Table 8 (Users Table)

Field Name	Data Type	Description (Optional)
UserID	AutoNumber	The ID of the user
UserName	Short Text	The name of the user
LastPlay	Date/Time	The last time this user loaded their profile
XP	Number	The XP points of the user
Health	Number	The user's default health
Damage	Number	The user's default damage
FireRate	Number	The user's default fire rate
Speed	Number	The user's default speed
NumDeaths	Number	The number of times the user has died
NumKills	Number	The number of times the user has killed an entity
NumQuestsCompleted	Number	The number of quests the user has completed
NumTimesHealthUpgraded	Number	The number of times the user has upgraded their health
NumTimesDamageUpgraded	Number	The number of times the user has upgraded their damage
NumTimesFireRateUpgraded	Number	The number of times the user has upgraded their fire rate
NumTimesSpeedUpgraded	Number	The number of times the user has upgraded their speed

Text File 1 (Map Tile Segment File)

```
This text file contains 5x5 grids of tile-IDs that describe each tile map segment

1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1

2 2 2 2 2
2 2 2 2 2
2 2 2 2 2
2 2 2 2 2
2 2 2 2 2

2 2 2 2 2
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
2 2 2 2 2

2 1 1 1 2
2 1 1 1 2
2 1 1 1 2
2 1 1 1 2
2 1 1 1 2
```

Image File 1 (Character Sprite Sheet)

Image File 2 (Item Sprite Sheet)

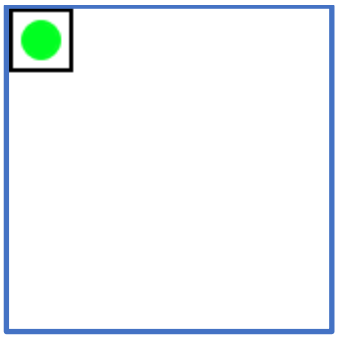


Image File 3 (Projectile Sprite Sheet)

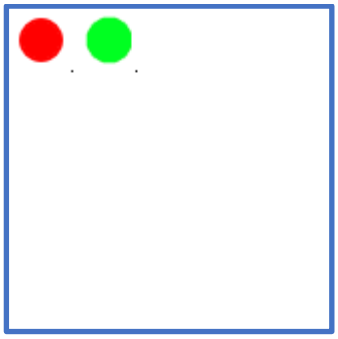
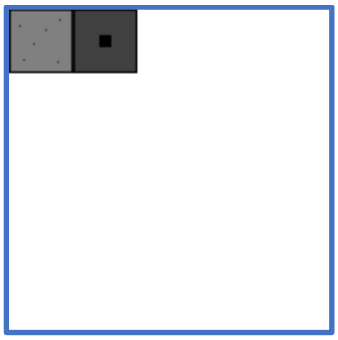


Image File 4 (Tile Sprite Sheet)



Explanation of Storage Design

File	Explanation
Table 1	The list of all active quests can be constantly changing with new quests being inserted and old quests being deleted, thus a database would be more appropriate than a database. There is a lot of data that needs to be stored and it is all of the same format, so a database table would be appropriate.
Table 2	Lots of enemy types with many attributes must be stored, so a database would make it easier to retrieve the data.
Table 3	A lot of items need to be stored with lots of attributes for each item, so a database table is more appropriate than a text file.
Table 4	A lot of projectile types must be stored. Retrieving all necessary data will be easier with a database table. The enemy damage types are linked to the projectile types so a database makes this relationship a lot easier to maintain than a text file.
Table 5	Each quest type can have multiple targets. A text file would make this very clumsy as the relationship between quest types and targets would not be convenient. Thus, a database is a lot more practical to use than a text file.
Table 6	The quest types must be linked to the quest targets table so a database allows for easier maintenance of this relationship. Thus, a database is a more appropriate choice than a text file.
Table 7	The data for each tile is in the same format and there can potentially be a lot of tiles added. Thus, a database would be easier to use than a text file.
Table 8	A lot of data needs to be stored about the users and all of it is in the same format. Therefore, a database would be more appropriate than a text file.
Text File 1	The data being stored is not related to the other values being stored so a database is not necessary. The data being stored is very simple and will only ever be read, not written to, so a database is also not necessary. Thus, a text file satisfies all of the requirements for this file and is thus a better choice than a database.
Image File 1	The .png format is a standard format for images. An image file is a lot easier to edit than images stored in a database. Thus, a regular image file is more appropriate than a database.
Image File 2	The .png format is a standard format for images. An image file is a lot easier to edit than images stored in a database. Thus, a regular image file is more appropriate than a database.
Image File 3	The .png format is a standard format for images. An image file is a lot easier to edit than images stored in a database. Thus, a regular image file is more appropriate than a database.
Image File 4	The .png format is a standard format for images. An image file is a lot easier to edit than images stored in a database. Thus, a regular image file is more appropriate than a database.