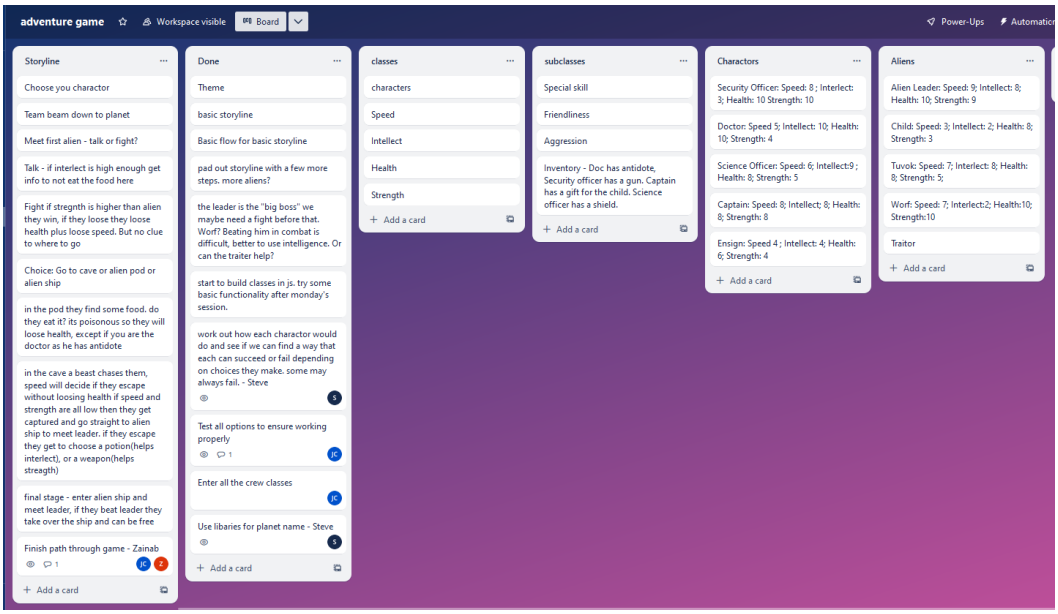
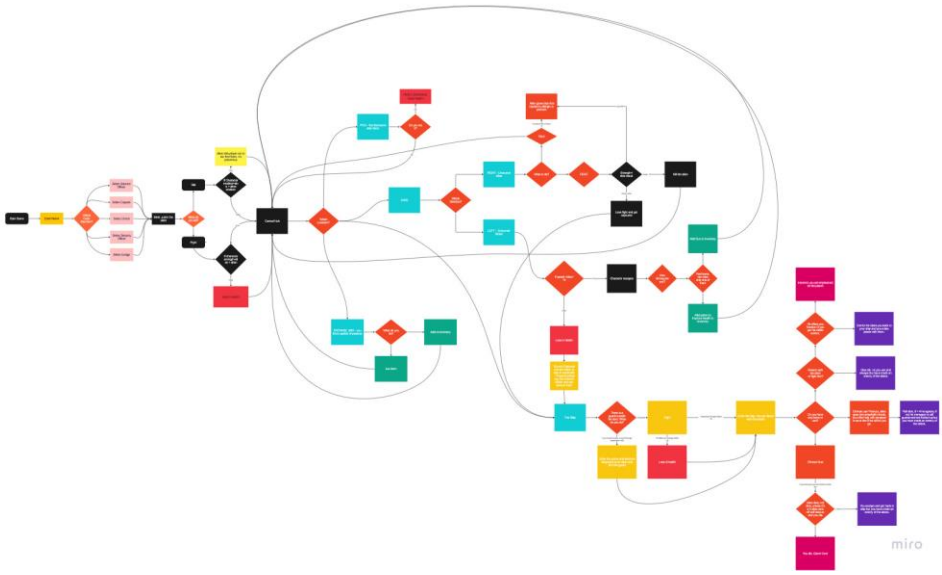
Text Based Adventure Game Assignment

This assignment was designed to develop skills in Object Oriented Programming and introduce NPM libraries. Student's were split into teams.

The first part of the task was to decide on a theme and plan the structure of the game. Trello and Mira were suggested as suitable tools for this.

Space was chosen as our theme. We then moved onto planning.





Having decided on the theme and mapped out a plan with a flowchart we moved onto the coding.

First up would be defining the classes (shown below is the final code, some alterations were made dring the coding stage to improve the flow.

```
export class Character{
    constructor (name, intellect, strength, speed,
        health){
        this.name = name;
        this.intellect = intellect;
        this.strength = strength;
        this.speed = speed;
        this.health = health;
        }
        stats(){
            return console.table({
                name : this.name,
                intellect: this.intellect,
                strength: this.strength,
                speed: this.speed,
                health: this.health,
                skill: this.skill,
                inventory: this.inventory
            });
```

With subclasses for alien characters and the crew:

```
export class aliens extends Character {
        constructor (name, intellect, strength, speed,
            health, aggression) {
            super (name, intellect, strength, speed,
                health, aggression);
                this.aggression = aggression
    }
}
export class Junior extends aliens {
    constructor (name, intellect, strength, speed,
        health, aggression) {
        super (intellect, strength, speed,
            health, aggression);
        this.name = name
        this.intellect = 5
        this.strength = 2
        this.speed = 3
        this.health = 3
        this.aggression = 2
        }
```

```
export class crew extends Character {
    constructor (name, intellect, strength, speed,
        health,skill, inventory) {
        super (name, intellect, strength, speed,
            health, skill,inventory);
            this.inventory = inventory;
    }
}
export class Doctor extends crew {
    constructor (name, intellect, strength, speed,
        health, inventory) {
        super (intellect, strength, speed,
            health, inventory);
        this.name = name
        this.intellect = 10
        this.strength = 5
        this.speed = 4
        this.health = 10
        this.skill = 'Cure'
        this.inventory = 'Empty'
    }
```

Having set these up we moved onto the gameplay. As shown in the flowchart the first stage would be to choose a character and establish them from the classes.

```
const start = async () => {     //choose your charcater - using the classes and
sub-classes get out in classes.js
    console.clear()
    console.log(`You arrive at planet ${randomP()} `.yellow)
    console.log('choose wisely')
    let character = await getcharacter()
    console.log(`You have chosen the ` + character)
    if (character == "Doctor") { chosen = new allOfThem.Doctor ('McCoy')
    crewimage = await crewpics.mccoyImage();
    console.log(crewimage.cyan)}
    if (character == 'Security Officer') { chosen = new allOfThem.securityOfficer
('Worf')
    crewimage = await crewpics.worfImage();
    console.log(crewimage.yellow)}
    if (character == "Science Officer") { chosen = new allOfThem.scienceOfficer
('Spock')
    crewimage = await crewpics.spockImage();
    console.log(crewimage.blue)}
    if (character == "Captain") { chosen = new allOfThem.Captain ('Kirk')
    crewimage = await crewpics.kirkImage();
```

```
    console.log(crewimage.yellow)}
    if (character == "Ensign") { chosen = new allOfThem.Ensign ('Crusher')
    crewimage = await crewpics.crusherImage();
    console.log(crewimage.red)}
```

As can be observed from the code some styling has been added, this was done later in the development and will be explained later.

To link all the js files together exporting and importing was required. As the program grew further files and connections were added until we had this:

```
import {getcharacter} from "./game.js"
import {alien, gameover, gamora, leon, winner, podImage, contImage,
mathesarImage, beastImage} from "./alienimage.js"
import {hubpic} from "./alienhubimage.js"
import {cavepic} from "./caveimage.js"
import  * as allOfThem from "./classes.js"
import * as allMeets from "./meetJ.js"
import * as crewpics from "./crewimages.js"
import { location } from "./location.js"
import {pod} from "./pod.js"
import {storageA, storageB, storageC} from "./storage.js"
import * as caveOptions from "./cave.js"
import {rightTunnel} from "./cave.js"
import {entrance, inventuse} from "./sEntrance.js"
import {interior, invtest, interior2, tryagain} from "./sInterior.js"
import randomItems from "planet-names";
import uniqueRandomArray from 'unique-random-array';
import {riddle} from "./newRid.js"
```

The first alien encounter was created from the “Junior” subclass. As with most of the encounters the played would be given a fight or talk option and the decision would then play out, again using classes to influence the outcome.

The interaction was handled by the enquirer module, an example is shown below:

```
import inquirer from "inquirer";
export const meetJunior = async(alien) => {
    let {meetJ} = await inquirer.prompt({
        message: `Oh no you have come across your first alien, ${alien} You must
make the decision to FIGHT or TALK.`,
        type: 'list',
        choices: ['TALK','FIGHT'],
        name: "meetJ"
    } )
```

```
    return meetJ
```

The interactions were handled in the main script file.

```
if (meeting1 == 'FIGHT') {
           console.log('You are going to fight the alien')      // can you beat
him?
           console.log(chosen)
            let round1 = chosen
            round1.fight(alien1.name)

    let meeting2= await allMeets.meetJunior2(alien1.name)     // the fight
continues
    if (meeting2 == 'FIGHT') {
        console.log(`You are going to continue fighting ${alien1.name}?`)
        console.log(chosen)
         console.log('You are distracted by when the aliens radio starts playing'
+ ' Never gonna give you up by Rick Astley'.blue + ' and the alien hits you');
chosen.health = chosen.health - Math.floor((Math.random()*4) + 1)
         if (chosen.health > alien1.health)
         {console.log(`${alien1.name} is beat, he gives up and says: spare me and
I'll give you tell you something important`)
         let meeting3= await allMeets.meetJunior3(alien.name)
         if (meeting3=='SPARE HIM'){console.log(`${alien1.name} tells you that
the Alien Leader is allergic to peanuts.  You smile as you leave him to
recover`)}
        else {console.log(`You kill ${alien.name} in cold blood, you will learn
nothing from him now, what a waste`)}
        }
         else {console.log(`You run away fron ${alien1.name} while you still
can`)}
```

The results were dealt with by classes with the outcome affected by the crew member chosen.

```
    this.name = name
        this.intellect = 3
        this.strength = 10
        this.speed = 8
        this.health = 10
        this.skill = 'Body Armour'
        this.inventory = 'Empty'
    }
    fight(alienbeing) {
        ;
```

```
        console.log(`${alienbeing}s punch against ${this.name} has little affect
due to his ${this.skill}`); this.health -=1;
        this.stats();
        return this;
    }
    badFood() {
        if (this.inventory == "Antidote") {this.inventory = "Empty";
console.log(`Thankfully ${this.name} has an antidote to the poison in his
inventory`);}
        else { this.health -=4; console.log(`${this.name} didn't like the taste
of that and starts seeing goblins running around him, getting trippy man, loose 2
health.`)};
     this.stats();}
```

Later in the game the player has the option to go to a storage unit and get an item. Depending on decisions made elsewhere the player may or may not know that the leader is allergic to peanuts and that the mushrooms are poisonous unless you have the cure (like the Doctor) or an antidote. Here is the code for the storage unit:

```
const mainStorage = async (storage) => {
    let contpic = await contImage ();
        console.log(contpic.magenta)
    console.log('You enter the storage unit')
    console.log(antistore,peantusstore)

    if (antistore==1 && peantusstore==1 ) {console.log('There is nothing in
here')}
    if (antistore==0 && peantusstore==0 ) {storage = await storageA();
console.log(storage)}
    if (antistore==1 && peantusstore==0 ) {storage = await storageB();
console.log(storage)}
    if (antistore==0 && peantusstore==1){storage = await storageC();
console.log(storage)}
    if(storage.charAt(0)== 'E'){
        console.log('You have chosen to eat them, tasty!'); chosen.health += 2;
console.log('Your health is now ' + chosen.health);
        peantusstore=1}
    if(storage.charAt(0)== 'R'){
        console.log('You have chosen to put them in your inventory');
chosen.inventory = "Peanuts"; console.log('Your inventory now has ' +
chosen.inventory + ' in it.');
        peantusstore=1}

    if(storage.charAt(0)== 'P'){
```

```
        console.log('You have chosen to put it in your inventory');
chosen.inventory = "Antidote"; console.log('Your inventory now has ' +
chosen.inventory + ' in it.');
        antistore=1}
    if(storage.charAt(0)== 'L'){
            console.log('You have chosen to stick with your current inventory')};

    console.log(`You head back the hub`.blue)
    setTimeout(hub, 2000)}
```

And the functions iit calls:

```
export const storageA = async() => {
    let {inventory} = await inquirer.prompt({
        message: 'You have just entered the storage unit and have found a packet
of peanuts and some kind of antidote. What would you like to do with them?',
        type: 'list',
        choices: ['Eat the peanuts', 'Replace anything you have in inventory with
the peanuts',
                    `Pick-up the antidote and replace anything you may have in
your inventory with it`,
                    'Leave the items and exit the unit'],
        name: "inventory"
    } )
    return inventory
}
export const storageB = async() => {
    let {inventory} = await inquirer.prompt({
        message: 'You have just entered the storage unit and have found a packet
of peanuts. What would you like to do with them?',
        type: 'list',
        choices: ['Eat the peanuts', 'Replace anything you have in inventory with
the peanuts', 'Leave the item and exit the unit',
                    ],
        name: "inventory"
    } )
    return inventory
}
export const storageC = async() => {
    let {inventory} = await inquirer.prompt({
        message: 'You have just entered the storage unit and have found an
antidote. What would you like to do with it?',
        type: 'list',
        choices: ['Pick-up the antidote and replace anything you have in your
inventory with it', 'Leave the item and exit the unit',
```

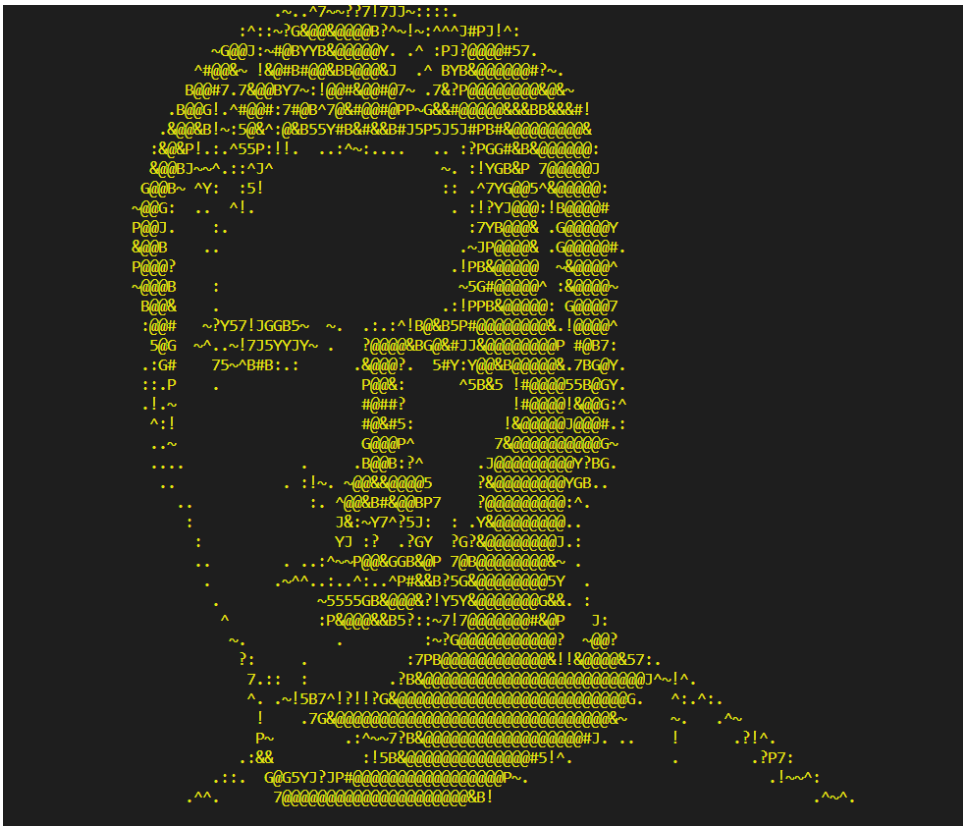
```
                    ],
        name: "inventory"
    } )
    return inventory
}
```

It got a bit more complicated once we put 2 items in the unit that, once used, are removed from the options. The method used is a bit clunky, a more streamlined way to do it would be to use an array.

Once the structure and the interactions were completed, we moved on to testing the game for bugs and to ensure a player could win or lose depending on the choices they took. As a game it is quite short and it is possible to complete it quite quickly so we may need to tweak it to make it more difficult.

The final task was to add some style with a colour text library and any other features we could devise to make it look more interesting. We added a “colors” package to do this, plus a random planet generator to choose the planet of the adventure. Remembering what text adventures really did look like we also added some images using an online converter all called them at various points within the game. To improve the flow, we added time interval commands to allow players to read the text before the next part was shown. This caused some issues as the function would continue to run and cause undesirable effects or bugs. We did try a random riddle generator but where not able to get it functioning with the time we had to put in a single riddle.

Here is an example of the images created:



Finally, if you win the game it will congratulate but remind you of the poor aliens you killed, if you chose the more aggressive route by pushing their names into an array.

Testing took a long time, with the timeouts for delays causing the most problems.

Here is a link to the repository created in GitHub:

https://github.com/DataSpot42/textGame

Here are some links to the resources used:

Image to ascii convertor:

https://www.text-image.com/convert/ascii.html

NPM colors package:

https://www.npmjs.com/package/colors

NPM inquisitor:

https://www.npmjs.com/package/inquisitor

NPM Planet-names:

https://www.npmjs.com/package/planet-names

Trello for workspace collaboration:

https://trello.com

Miro Tool for workflow collaboration:

https://miro.com