

(GAD170 - 24T1) - Pseudo Code - Project 2

Gamemode Manager Script -

Variables:

// boolean for letting the game know when the game is over

Declare isGameOver bool = false

// references to the other 2 classes needed this class is the only class with references to the other classes in order for all classes to funnel properly to the main class which is the gamemode manager

Declare npcManagerRef NPCManager class

Declare uiManagerRef UIManager class

// crew member list which holds the information of each crew member recruited by the player

Declare activeCrewMembers List string firstName, string lastName, string hobby

Functions:

// gets all references needed for the class to work properly

GetAllReferences()

npcManagerRef = GetComponent of type NPCManager class

uiManagerRef = GetComponent of type UIManager class

// adds a new human member to the list by inputting the first name, last name & hobby of the recruited members portfolio when the player recruits the member

AddNewCrewMemberToList(string firstName, string lastName, string hobby)

Get firstName, lastName, hobby parameter values & add to activeCrewMembers list

// code executed when the player clicks the recruit button on a crew member portfolio

RecruitCrewMember()

If npcManagerRef of type chosenHobby = any 4 alien hobbies of alienHobbies list

Local string choseHobbyToAttack = random human hobby from npcManagerRef of type humanHobbies

Call RemoveInfoFromCrewBoard with input parameters (choseHobbyToAttack local var) from uiManager class

// local list similar to the active crew members list but is used to collect all the crew members to remove for later, this will add the members to be removed to a local list & in another for each loop will remove the active crew members based on the local list

Local membersToRemove list of type tuple string firstName, string lastName, string hobby

// uses a for each loop & adds the crew member which has the hobby targeted by the alien to the members to remove list

For each loop which loops through each member in activeCrewMembers list

if member of type hobby = chosenHobbyToAttack
Add current member to membersToRemove list

// for each loop is used to remove the members from the active crew members list using the members to remove list as the collection

For each loop which loops through each member in membersToRemove list

Remove current member from memberactiveCrewMembers list

Call ChooseNewCrewMember() from npcManager class
Call SetCrewPortfolio with input parameters (npcManagerRef of type generatedFirstName, npcManagerRef of type generatedLastName, npcManagerRef of type chosenHobby) from uiManager class

Else

Local string firstName = npcManagerRef of type generatedFirstName
Local string lastName = npcManagerRef of type generatedLastName
Local string hobby = npcManagerRef of type chosenHobby

Call AddNewCrewMemberToList with input parameters (firstName, lastName, hobby)
Call AddInfoToCrewBoard with input parameters (firstName, lastName, hobby) from uiManager class
Call ChooseNewCrewMember() from npcManager class
Call SetCrewPortfolio with input parameters (npcManagerRef of type generatedFirstName, npcManagerRef of type generatedLastName, npcManagerRef of type chosenHobby) from uiManager class

// code executed when the player clicks the reject button on a crew member portfolio
RejectCrewMember()

Call ChooseNewCrewMember() from npcManager class
Call SetCrewPortfolio() with input parameters (npcManagerRef of type generatedFirstName, npcManagerRef of type generatedLastName, npcManagerRef of type chosenHobby) from uiManager class

Start()

// gets all references at the start of the game

Call GetAllReferences()

// execute markov chain name generator

Call MarkovFirstNameGenerator() from npcManager class

Call MarkovLastNameGenerator() from npcManager class

Call ChooseHobby() from npcManager class

Call SetCrewPortfolio with input parameters (npcManagerRef of type generatedFirstName, npcManagerRef of type generatedLastName, npcManagerRef of type chosenHobby) from uiManager class

Update()

// if the player collects a total of 10 crew mates & the game is not over than the portfolio widget & crew member board widget will vanish & a "You Win" text will show up

// the isGameOver bool will be set to true to this code will only be executed once

if activeCrewMembers list count = 10 & isGameOver = false)

 Print "You Win" to the log

 Set isGameOver to true

// sets portfolio widget to inactive

uiManagerRef of type portfolioRef sets SetActive to false

// for each crew member widget the loop will set each member widget to inactive

For each loop which loops through each member in crewMembers list from uiManager class)

 Member set SetActive to false

// sets the win text to active

uiManagerRef of type winObject set SetActive to true

UI Manager Script -

Variables:

// references to all the UI game objects

Declare portfolioRef GameObject variable

Declare profilePicRef GameObject variable

Declare firstNameTxtResultRef GameObject variable

Declare lastNameTxtResultRef GameObject variable

Declare hobbyTxtResultRef GameObject variable

// references to the list of crew member game objects text information

Declare firstNameTextMeshProUGUI variable
Declare lastNameTextMeshProUGUI variable
Declare hobbyTextMeshProUGUI variable

// references to the list crew member game objects image information

Declare profileImage Image variable
Declare full_Image Image variable

// reference to the win text game object

Declare winObject GameObject variable

// lists for the sprite character images, however there is only one item in each list so it is only here if I decided to use more than one sprite character

Declare profileCharacterImage List Sprite
Declare crewCharacterImage List Sprite

// reference to active crew member list information

Declare crewMembers List GameObject

// dictionary which assigns the full length sprite image (active crew member on team) to the half image (portfolio image)

Declare crewImages Sprite, Sprite dictionary

Functions:

// adds all the references to the image & text components

AddReferences()

// references to portfolio information

profileImage = profilePicRef.GetComponent of type Image
firstName = firstNameTxtResultRef.GetComponent of type TextMeshProUGUI
lastName = lastNameTxtResultRef.GetComponent of type TextMeshProUGUI
hobby = hobbyTxtResultRef.GetComponent of type TextMeshProUGUI
winObject = transform.Find name of object (WinTxt) of type gameObject

// adds the images in the lists to the dictionary, this is done by assigning a full length version of an image to the portfolio key image

AddToCrewImageDictionary()

For loop which loops through as many times as crewCharacterImage length value
 profileCharacterImage[at index], crewCharacterImage[at index] added to crewImages dictionary

// set crew member portfolio information

SetCrewPortfolioImage()

profileImage of type sprite = profileCharacterImage at first index

// sets the crew portfolio to either be accepted or rejected

SetCrewPortfolio(string firstNameInput, string lastNameInput, string hobbyInput)

Call SetCrewPortfolioImage()

firstName of type text = firstNameInput parameter value

lastName of type text = lastNameInput parameter value

Hobby of type text = hobbyInput parameter value

// adds the information from the recruited members portfolio into the crew team board, this adds a crew member to the players team

AddInfoToCrewBoard(int crewID, string firstName, string lastName, string hobby)

For each loop which loops through every member in crewMembers tuple list

// gets a local variable reference to the prefab crew members first name (game object), last name (game object) & hobby (game object) based on the last crew member in the list

Local Image profilePic = member.GetComponent of type Image

Local GameObject firstNameRef = member.transform.Find object called FirstName

Local GameObject lastNameRef = member.transform.Find object called LastName

Local GameObject hobbyRef = member.transform.Find object called Hobby

// gets a local variable reference to the chosen crew members first name (text mesh), last name (text mesh) & hobby (text mesh)

Local TextMeshProUGUI firstNameTxt = firstNameRef.GetComponent of type TextMeshProUGUI

Local TextMeshProUGUI lastNameTxt = lastNameRef.GetComponent of type TextMeshProUGUI

Local TextMeshProUGUI hobbyTxt = hobbyRef.GetComponent of type TextMeshProUGUI

// if the crew member slot has there first name set to first name & not an actual name this means that the crew member slot is empty

if firstNameTxt of type text = First Name

// sets all the information for the recruited crew member

profilePic of type sprite = crewCharacterImage at first index

firstNameTxt of type text = firstName

lastNameTxt of type text = lastName

hobbyTxt of type text = hobby

Break from for each loop

// removes members from the players team if an alien is recruited, the members removed will be based there hobby & which hobby is chosen by the alien to attack

RemoveInfoFromCrewBoard(string hobby)

For each loop which loops through every member in crewMembers tuple list

```
Local Image profilePic = member.GetComponent of type Image
Local GameObject firstNameRef = member.transform.Find object called FirstName
Local GameObject lastNameRef = member.transform.Find object called LastName
Local GameObject hobbyRef = member.transform.Find object called Hobby
```

```
Local TextMeshProUGUI firstNameTxt = firstNameRef.GetComponent of type
TextMeshProUGUI
Local TextMeshProUGUI lastNameTxt = lastNameRef.GetComponent of type
TextMeshProUGUI
Local TextMeshProUGUI hobbyTxt = hobbyRef.GetComponent of type
TextMeshProUGUI
```

```
if hobbyTxt of type text = hobby parameter value
```

```
    profilePic of type sprite = null
    firstNameTxt of type text = "First Name" string
    astNameTxt of type text = "Last Name" string
    hobbyTxt of type text = "Hobby" string
```

Start()

// calls the add functions for dictionaries & references

```
Call AddToCrewImageDictionary()
```

```
Call AddReferences()
```

// debugs the win text objects name for debugging purposes only in order to make sure I had the correct reference

```
Print winObject of type name to log
```

// sets the win text object to inactive at the start of the game

```
Set winObject active to false
```

NPC Manager Script-

Variables:

// declare current letter variable used for name generation

```
Declare currentLetter char variable
```

// declare first name variables

```
Declare generatedFirstName string variable
```

```
Declare firstNameLength int variable
```

```
Declare firstNameMinLength int variable
```

```
Declare firstNameMaxLength int variable
```

// declare last name variables

Declare generatedLastName string variable

Declare lastNameLength int variable

Declare lastNameMinLength int variable

Declare lastNameMaxLength int variable

// declare chosen random hobby variable

Declare chosenHobby string variable

// declare dictionaries for markov first name & last name generator

Declare firstNameDictionary char, list char, float dictionary

Declare lastNameDictionary char, list char, float dictionary

// declare starting letter for first name & last name

Declare firstNameStartLetterList char list

Declare lastNameStartLetterList char list

// list of hobbies to choose from for the human & alien members

Declare humanHobbies string list

Declare alienHobbies string list

Functions:

// add chars to first letter of name list

AddCharToFirstNameList function()

Clear the firstNameStartLetterList

Create new firstNameStartLetterList with chars used for start of a name

// add chars to first letter of name list

AddCharToLastNameList function()

Clear the lastNameStartLetterList

Create new lastNameStartLetterList with chars used for start of a name

// add to hobby list

AddToHumanHobby()

Add range of human hobbies as string to human hobbies list

// add to hobby list

AddToAlienHobby()

Add range of human hobbies as string to alien hobbies list

// add letter values with probability values assigned to each letter in the list

// highest probability char value in list is first in the list for organization

AddToFirstNameDictionary()

Add char (key) & list of chars (letter), floats (probability) to firstNameDictionary dictionary

// add letter values with probability values assigned to each letter in the list

// highest probability char value in list is first in the list for organization

AddToLastNameDictionary()

Add char (key) & list of chars (letter), floats (probability) to lastNameDictionary dictionary

MarkovFirstNameGenerator()

// sets the generated first name & current letter to empty

generatedFirstName = empty

currentLetter = empty

// sets current char to a char from first name letter list, then adds it to the first index of the generated name

currentLetter = random char chosen from the firstNameStartLetterList list

currentLetter is added to generatedFirstName string

// sets the min - max first name length range

firstNameMinLength = 4

firstNameMaxLength = 7

// sets the length of the name with true max name length

firstNameLength = a random int between firstNameMinLength & firstNameMaxLength

// loop dictionary to create name based on name length

For loop which loops through as many times as the firstNameLength value

LoopFirstNameDictionary

// makes the first letter of the fully generated first name a capital letter

generatedFirstName = (cast as char) generatedFirstName[0] - 32) +

generatedFirstName.Substring(1)

MarkovLastNameGenerator()

// sets the generated last name & current letter to empty

generatedLastName = empty

currentLetter = empty

// sets current char to a char from last name letter list, then adds it to the first index of the generated name

currentLetter = random char chosen from the lastNameStartLetterList list

currentLetter is added to generatedLastName string

// sets the min - max last name length range

lastNameMinLength = 4

lastNameMaxLength = 8

// sets the length of the name with true max name length

lastNameLength = a random int between lastNameMinLength & lastNameMaxLength

// loop dictionary to create name based on name length

For loop which loops through as many times as the lastNameLength value

LoopLastNameDictionary

// makes the first letter of the fully generated last name a capital letter

generatedLastName = (cast as char) generatedLastName[0] - 32) +

generatedLastName.Substring(1)

LoopFirstNameDictionary()

// checks if the current letter has a spot in the dictionary, if not then the loop is done

if firstNameDictionary contains the currentLetter char value

// sets the local "value" to the tuple list of the current letter

List char (letter), float (probability) value = firstNameDictionary [currentLetter]

// local "randomValue" gets a random float between 0.0 & 1.0

float randomValue = random float between 0.0, 1.0

// null check for error prevention

if value doesn't = null

if value probability float at index 0 = 1.0

Value.letter at [0] is added to generatedFirstName

Else if value probability float at index 0 = 0.5

**// the local "decider" var will get a random value of 0 - 1 which will
equally decide randomly which tuple will be chosen**

int decider = random int between 0 or 1

// if the local "decider" var is equal to 0 choose tuple 1

If decider = 0

Value.letter at [0] is added to generatedFirstName string

// if the local "decider" var is equal to 1 choose tuple 2

Else

Value.letter at [1] is added to generatedFirstName string

Else if value.probability at [0] is greater than local randomValue variable

Value.letter at [0] is added to generatedFirstName string

Else

Value.letter at [1] is added to generatedFirstName string

// gets the last letter of the name & sets it to the current letter

currentLetter = generatedFirstName [last index]

LoopLastNameDictionary()

// checks if the current letter has a spot in the dictionary, if not then the loop is done

if lastNameDictionary contains the currentLetter char value

// sets the local "value" to the tuple list of the current letter

List char (letter), float (probability) value = lastNameDictionary [currentLetter]

// local "randomValue" gets a random float between 0.0 & 1.0

float randomValue = random float between 0.0, 1.0

// null check for error prevention

if value doesn't = null

if value probability float at index 0 = 1.0

Value.letter at [0] is added to generatedLastName

Else if value probability float at index 0 = 0.5 then true branch is executed

**// the local "decider" var will get a random value of 0 - 1 which will
equally decide randomly which tuple will be chosen**

int decider = random int between 0 or 1

// if the local "decider" var is equal to 0 choose tuple 1

If decider = 0

Value.letter at [0] is added to generatedLastName string

// if the local "decider" var is equal to 1 choose tuple 2

Else

Value.letter at [1] is added to generatedLastName string

Else if value.probability at [0] is greater than the local randomValue variable

Value.letter at [0] is added to generatedLastName string

Else

Value.letter at [1] is added to generatedLastName string

// gets the last letter of the name & sets it to the current letter

currentLetter = generatedLastName [last index]

SetupCrewMember()

// setup char list & dictionary for first name

Call AddCharToFirstNameList()

Call AddToFirstNameDictionary()

// setup char list & dictionary for last name

Call AddCharToLastNameList()

Call AddToLastNameDictionary()

// setup list of hobbies for humans & aliens

Call AddToHumanHobby()

Call AddToAlienHobby()

// chooses a random hobby for the crew members portfolio

ChooseHobby()

chosenHobby = empty

// setup chosen crew member hobby

Int isAlien = random int between 0 or 1

// if crew member is alien than the if branch will be chosen, if not than the else statement will be chosen

If isAlien = 0

chosenHobby = random hobby from alienHobbies list

Else

chosenHobby = random hobby from humanHobbies list

ChooseNewCrewMember()

// calls name generator & hobby functions

Call MarkovFirstNameGenerator()

Call MarkovLastNameGenerator()

Call ChooseHobby()

Start()

// generates the crew members first name & last name

SetupCrewMember()