Jessica Albert
Final Project
CS 162 W'19

Game Description – Game of Life – Twisted Path Edition

I decided to write a game reminiscent of what I'm sure many people in this program are traversing through right now. Most of us are career changers that already have a degree in a different field and our goal is to break into software development and obtain a job in that field, thus we are on what one would call a "Twisted Path".

In the game the Player moves from space to space to gain Technical, Courage, Confidence points while losing Doubt points. Typically when people dive into a career change they start out with a lot of doubts, a good amount of courage, a low amount of confidence, and a really low amount of technical skills. This is reflected in the Player's starting stats. The objective of this game is to gain a Job within 20 steps.

There are 3 types of Space objects (inherited from the abstract Space class, explained below): Study, Project, and Apply. The Board class generates and links a grid of these space objects together. The Board class is also totally scalable in that you can change the rows, columns, and number of Space types in the code. In my implementation I created a board of 16 Spaces (13 Study, 2 Project, 1 Apply).

Each space provides different benefits and potential setbacks to the Player. The Study Space adds study points to the Player's stats: 2 study points allows the player to complete a class which adds 1 Technical, 1 Confidence and lose 1 Doubt. If the player completes 8 classes they earn a Degree object (Items discussed below). The setback for the Study class is a challenging class which causes the player to lose a Confidence point. The Project Space has no setbacks and allows the Player to earn a Personal Project Item which helps them to get the Job (the objective). Finally the Apply space has a 50/50 chance of resulting in a setback. If a setback happens the player gains a Doubt and loses a Confidence. If a positive outcome occurs the Player's application is checked to make sure that they have 0 Doubt points, 5 Technical, 5 Courage, 5 Confidence and at least 1 Personal Project Item. If all these check out the Player get's a job and the objective is achieved.

Setbacks/positive outcomes are determined by asking the user to select a number 1 – the Space object's "max sides" (this represents the probability of getting a setback outcome). If the result of the "roll" matches the user's selection a setback occurs in the round (details below). The Study space setback is a 1 in 5 chance, Apply is a 1 in 2 chance.

The Game class ties together the Player and Board object and contains all the gameplay flow functions. This includes the round menu, function to move the Player around the Board, display the available moves, stats, Items, etc.

The Player has a vector of pointers to Items objects which represents objects in their toolkit. The Items class is also an abstract class, the following classes inherit from it: Personal Project, Professional Contact, Reference, Degree, and Job. In the current implementation of the game gaining a Professional Contact gives the player the chance to also gain a Reference (1 in 10 chance). If this happens they Player is fast tracked to achieving the objective and they gain a Job object. This ends the game. The player needs a Personal Project to get past the application check.

Jessica Albert
Final Project
CS 162 W'19

Working Notes

1. Game of Life – Twisted Path Edition
    a. Goal is to get a job in computer science
    b. Only have 20 steps to gain all necessary points to get job
    c. Each step requires a roll of a die, if the result matches a randomly generated number then a setback on your journey occurs
    d. Shed doubts, gain confidence, courage, and technical know-how to land a job as a programmer.
    e. Starting stats
        i. 5 doubt
        ii. 4 courage
        iii. 1 confidence
    f. Space events
        i. 2 Project – gain technical, confidence, courage, shed doubt
            1. no setbacks possible
            2. items possible
                a. personal project
            3. action
                a. roll dice for setback or positive
                b. change numbers of sides to change how likely it is to fail
        ~~ii. 16 work – gain money, gain doubts~~
            ~~1. Setbacks possible~~
                ~~a. praise at work~~
            ~~2. Gain money each time worked - $4000 per work~~
        iii. 6 study – gain confidence, ~~indirectly~~ gain technical, shed doubt
            1. setbacks possible –
                a. challenging class
            2. positive outcome
                a. add 1 study point
            3. items possible
                a. networking event
                b. professional contact
        iv. 1 apply - can only choose this if 5 confidence, 5 technical, 5 courage
            1. Possible to have set back event if no personal project
                a. gain doubt, lose confidence
            2. Or goes well
                a. shed doubt, unlocks Job event (bool flipped to true within apply space)
    g. Key events
        i. Complete classes
            1. study 2 times
                a. gain 1 technical
                b. gain 1 confidence

c. shed 1 doubt

ii. job - objective

1. can only achieve if 5 courage, 5 confidence, 5 technical, 0 doubts

h. Secondary events

i. Challenging class gives you doubt

1. lose 1 confidence

ii. Networking event – potential to find professional contact

1. shed 1 doubt
2. gain 2 confidence

iii. Application setback

1. gain 1 doubt
2. lose 1 confidence

i. Items

i. personal project

1. gain 1 technical
2. shed 1 doubt
3. gain 2 confidence

ii. professional contact – very helpful for reference

iii. reference – add check to all places where professional contact gained (1 in 10 chance)

1. Can only get reference if personal project is completed and professional contact is made
2. Fast track to job objective, win game

iv. Degree – analyze in completeClass function

1. only if study points >=8

v. Job

j. Determine setback or positive

i. user picks number 1-6, roll dice, if match addsetback if no match positive

<u>Necessities for Program</u>

1. Grid of 20 Space class objects, linked via Space * pointers (up, down, left, right)
   a. How to change Space to inherited??? →use random number generator to create and link different inherited Space items to build the board. (Space can't be initialized…virtual)
2. Space Class with 4 inherited classes – pure virtual
   a. data members
      ~~i. x_pos~~
         ~~1. if x = 0, left = NULL **If NULL left not a move option~~
         ~~2. if x = 19, right = NULL~~
      ~~ii. y_pos~~
         ~~1. if y = 0, up = NULL~~
         ~~2. if y = 19, down = NULL~~
      iii. Up, down, left, right Space* pointers

iv. bool playerOccupies
    1. flip to true when player moves into space, false when player moves out of space
v. char symbol
vi. string type

b. functions
i. return space type
ii. setback
iii. positive
iv. random number
v. get random event
vi. display menu

3. Player class
a. data members
i. int confidence
ii. int doubt
iii. int courage
iv. int technical
v. vector items – need capacity and resize option
~~vi. int x_pos~~
~~vii. int y_pos~~
viii. int steps

b.

c. functions
i. move to new space
ii. display items – move to Item class
iii. display all data members
iv. add item

4. Game Class – include in reflection, encapsulation of gameplay functions
a. data members
i. Player
~~ii. Grid of spaces~~ Initialize board in separate ~~function~~ Board class, pass pointer to Player starting position instead
iii. number of each kind of space (to make scalable)
iv. Space *playerSpace = space the player is in

b. functions
i. initialize board
    1. 9 x 9 Spaces
    2. set Study (6)
    3. set Apply (1)
    4. set Project (2)
ii. print board
    1. space getChar

   iii. getMove
      1. print the type of space here
   iv. randomNumber
   v. randomEvent
   vi. moveMenu
   vii. roundOutcome
   viii. getInteger()
   ix. isInteger()
   x. setPlayerSpace()
      1. moves the Player pointer to the space selected by player
      2. updates playerOccupies bool for old space and new space
5. Board Class – to link spaces together
6. main menu
   a. move
   b. view stats
   c. display items
   d. quit

Jessica Albert
Final Project
CS 162 W'19

Test Table

Test Goal	Inputs	Expected Outcome	Code Changes
Build board of Space objects w/o memory leaks	Start with Board of 16 Study spaces, test with different variations of Space class objects as they are built	-No memory leaks -Board prints with 'S' in a grid of 4x4 (i.e Space is not initialized)	-Memory leak caused by "garbage" pointer in destructor not being set to "ptr" pointer in the Board destructor -Memory leak caused by issues with randomly generating Space types, fixed by adjusting min max for randomNumber() function -generating same board every time, needed to seed rand() member function in main()
Study Inherited class	build board of Study Objects, call member functions to see setbacks/positive outcomes	-Player stats updated accordingly Player Items update accordingly	non-as expected
Game menu, move menu only displays valid moves	Select menu function to move spaces. In first space only down and right are valid	-Only down and right are displayed, if select up or left display error and wait for user to select valid move	-none, as expected
Game menu, view stats and items	Wait until Player has obtained a couple items	-Stats and items are displayed accurately	-forgot to add a name data member for Inherited Items classes
Items vector properly adds items to Player Items pointer vector, printItems() function works properly	Play Project and Study spaces until a Personal Project and/or Degree and/or Professional Contact	List of items print by name	-Need to work through each Inherited Item and Space class file and make sure they include the correct files, lots of errors
Reference fast tracks to objective achieved	"hard code" reference to be obtained	Once Reference object obtained, Job constructor called, Game congratulates user and terminates	none, as expected

Jessica Albert
Final Project
CS 162 W'19

Game loop terminates when Player reaches 20 steps, tells user why game ended	Move though w/o hitting Apply step until 20 steps taken	Displays game is over and tell user why	-Changed <= 20 to < 20so that game would terminate at the correct time
Game loop terminates when Player quits	Select "Quit' in round menu	Displays game is over and tell user why	none- as expected
Apply class validates player stats required items to achieve objective	-Apply space early, hard code positive outcome -Apply to space when stats/items are good	-checkApplication will prevent objective from occurring -checkApplication allows objective to be achieved	-Doubts == 0 was changed to <= 0

Jessica Albert
Final Project
CS 162 W'19

Reflection

Overall I really enjoyed working on this project. It was very challenging because there was very little direction, however I felt that I was able to effectively plan and execute a well rounded game and theme. When the project was first released I was very nervous about having to come up with my own theme and implement the game with pretty much total freedom. However I was able to sit down and break down the game into little, more manageable pieces to work on and eventually I came out with a finished product. This definitely speaks to how far I have come since the beginning of this class. I did not think it was possible for my code to come this far in 10 weeks but I have definitely proven myself wrong with this project.

When I first planned the game I did not initially plan on having a Board or Game class, but I realized that these 2 things should definitely be encapsulated because they are distinct from each other and from the other classes I was implementing. Another thing that was my goal for the Board class was to be able to make the Board completely scalable, which I was ultimately able to achieve. This was a huge challenge and the most difficult part of the project for me because of the linked list structure of the Board. In previous projects we have always created a grid by implementing an array of Objects. However with the linked list structure I had to create an algorithm to move through the Board in a different way to build, print, and delete the Spaces in the Board because I could not rely on using the row/column indicators that an array would have.

Initially I thought that the Player object would have the ability to display it's x and y position on the Board, however as I started to figure out the way to traverse the board in a systematic way I realized that it would be easier to just print the Board to the user. I would say that I prefer the linked list structure for creating a grid of objects. Once I figured out how to build and link everything together I think it was a lot easier to move the player around the Board by using the next, prev, up, and down pointers.

I also decided to make the Player's object holder a vector of pointers to Items class objects. This was so that I could implement the Items class to be abstract as well and then create several inherited objects. While the inherited Items classes I built in this implementation were very simple, I felt that building it this way would allow me to expand on the project and make it more complex because each different Item can have special functions or different implementations of functions. This was definitely a breakthrough for me, in the beginning of the class I struggled to understand that the code I was writing was difficult to add features to. This was especially apparent in Project 4. I am glad that I have learned how to think beyond the project I am working on and write my code in a way that is scalable.