**OOP2 README- Ex. 4 Omar Abdel Al**

פרטי הסטודנטים: עמר עבד אלעל 316249911

הסבר כללי על התרגיל: התרגיל התעסק בתזכורת של הספרייה SFML. בתרגיל זה התבקשנו לממש משחק שנקרא 6 colors. על הלוח יש x מתומנים כאשר כל אחד צבוע באחד מששת הצבעים האפשריים. על המשתמש להצליח לצבוע כמה שיותר מתומנים בצבע שלו לפני השחקן השני (המחשב).

בתרגיל זה מימשנו 3 סוגים של שחקנים אויבים: אחד צבע אקראי, אחד שבוחר הצבע השחיח ביותר מבין השכנים, ואחד שבוחר צבע לפי מספר השכנים של השכנים.

תיכון:

Button.h- מחלקת הכפתורים של הצבעים שמאפשרים לשחקן לבחור צבע. כאן מופיעות ההצהרות על הפונקציות וההגדרה של המחלקה.

Button.cpp- מחלקת הכפתורים של הצבעים שמאפשרים לשחקן לבחור צבע. כאן מופיעים המימושים של הפונקציות.

Control.h- מחלקת ניהול המשחק. כאן מופיעות ההצהרות על הפונקציות וההגדרה של המחלקה.

Control.cpp- מחלקת ניהול המשחק. כאן מופיעים המימושים של הפונקציות.

Node.h- המחלקה שמייצגת את המתומנים על המסך של המשחק. כאן מופיעות ההצהרות על הפונקציות וההגדרה של המחלקה.

Node.cpp- המחלקה שמייצגת את המתומנים על המסך של המשחק. מופיעים המימושים של הפונקציות.

Resources.h- זוהי מחלקת singleton שטוענת את כל הקבצים החיצוניים הדרושים לאנימציה. כאן מופיעות ההצהרות על הפונקציות וההגדרה של המחלקה.

Resources.cpp- זוהי מחלקת singleton שטוענת את כל הקבצים החיצוניים הדרושים לאנימציה. כאן מופיעים המימושים של הפונקציות.

ScoreBoard.h- המחלקה שאחראית על שמירת הניקוד. כאן מופיעות ההצהרות על הפונקציות וההגדרה של המחלקה.

ScoreBoard.cpp- המחלקה שאחראית על שמירת הניקוד. כאן מופיעים המימושים של הפונקציות.

Toolbar.h- המחלקה האחראית על התפריט שמופיע מתחת למשחק. כאן מופיעות ההצהרות על הפונקציות וההגדרה של המחלקה.

Toolbar.cpp- המחלקה האחראית על התפריט שמופיע מתחת למשחק. כאן מופיעים המימושים של הפונקציות.

מבני נתונים עיקריים ותפקידיהם:

1. וקטור של Nodes שמייצג את לוח המשחק.
2. וקטור של טקסט ב ScoreBoard ששומר את הניקוד של השחקנים.
3. std::unordered\_map<std::shared\_ptr<Node>,std::list<std::shared\_ptr<Node>>> m\_adjLists;

זה רשימת השכנים של כל הקדוקדים

1. std::unordered\_map<int, void (Control::\*)()> m\_npcDifficultyMap;

זאת מפה שממפה בין רמת הקושי והפונקציה הדרושה

1. std::vector<std::shared\_ptr<Node>> m\_playerCluster;

זה וקטור ששומר את הקדקוקים של השחקן

1. std::vector<std::shared\_ptr<Node>> m\_NPCcluster;

זה וקטור ששומר את הקדקודים של המחשב

1. std::vector<std::shared\_ptr<Node>> m\_freeCluster;

זה וקטור ששומר את הקדקודים החופשיים

1. std::unordered\_map< std::string, sf::Color> stringToColorMap

זאת מפה שממפה בין ששם הצבע וערכו

1. std::unordered\_map<std::shared\_ptr<Node>, bool> visitedFreeNodes;

זאת מפה שעוזרת לנו לדעת איזה קדקודים חופשיים ביקרנו במהלך אלגוריתם החיפוש

אלגוריתמים הראויים לציון:

Function Control::createAdjLists()

creates adjacency lists for each node, when a node A intersects with node B and they both have the same color, then node B is added into node A adjacency list.

Control::updateNode()

Iterates through every player node held in m\_playerCluster and checks if it intersects any free node (kept in m\_freeCluster) and has the same color, is so then the node is removed from m\_freeCluster and added into m\_playerCluster then we go into the next function before recoloring all player nodes to players current color (this function does the exact same for the computer player)

Control::PushIntoAdjLists

the functions is called from the previous function, it receives as parameters the root or the node we want to start traversal from , and a type to know whether to add into m\_NPCcluster or m\_playerCluster, in each iteration over the adjacency list of the root we check if we have visited this current node using the map visitedFreeNodes, if so then we go to the next node, if we haven’t visited then we added to the right cluster(depending on the type passed to the function) then we take the free Node out of the freeCluster vector. At the the end we delete adjacency list of the passed root from m\_adjLists map.

easyNpc()

this function picks a random number between 1 and 6 and sets the m\_NPCcurColor according to the random number we generated.

mediumNpc()

this function uses a map to save how many nodes with a certain color is around the m\_NPCcluster nodes after we check all the surrounding nodes we iterate over the map again

and pick the most frequent color while making sure it is not the color the player picked

hardNpc()

iterates over free nodes surrounding the npc nodes and picks the color of the free node with the biggest adjacency list

באגים ידועים:

לפעמים בבחירה ראשונה של המחשב הוא לא מסמן איזה צבע הוא בחר אבל זה רק בפתיחת התוכנה במשחק הראשון

הערות:

אני מודע לכך שדחפתי מלא פונקציות וחישובים לתוך מחלקת ה control אבל אני אשמח לקצת התחשבות מכיוון שאני עובד לבד ובמקביל אני צריך לעבוד על פרויקטים בקורסים אחרים שגם הם כבדים