ASTRATEGOS

Proiect pentru examenul de atestat

Ursescu Sebastian

Petrescu Sorin

Vidrașc Sabin

CUPRINS

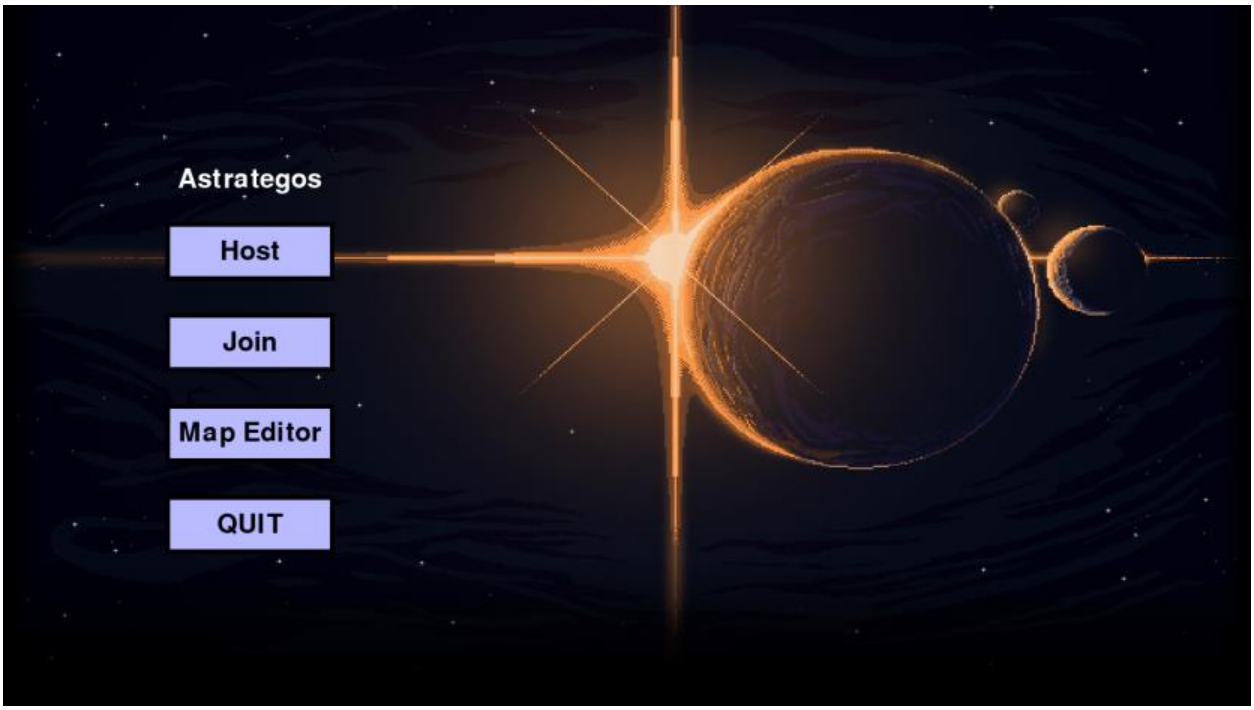
De ce am realizat acest proiect?

In cadrul acestui proiect, am ales să realizăm un joc de strategie bazat pe ture, introducând de asemenea și cateva aspecte din jocurile de strategie in timp real, cum ar fi explorarea nivelului, obținerea resurselor plasate strategic pe hartă și de asemenea se pune accent mai ales pe tactici și o strategie bună pentru a ieși invingător.

Astrategos este un joc de 2-4 jucători pe rețea pe calculatoare diferite. Un aspect important acestui joc este promovarea creativității, deoarece orice utilizator poate creea propriile sale nivele, un aspect fundamental al acestui proiect.

Structura proiectului

Odată cu intrarea in joc, utilizatorului îi sunt deja prezentate diferitele opțiuni pe care le poate alege.

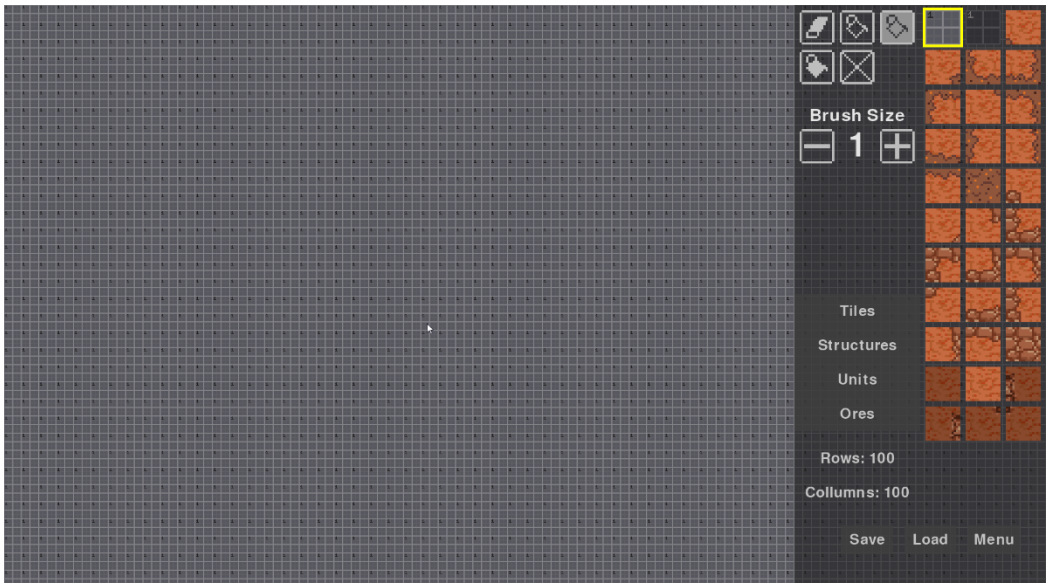


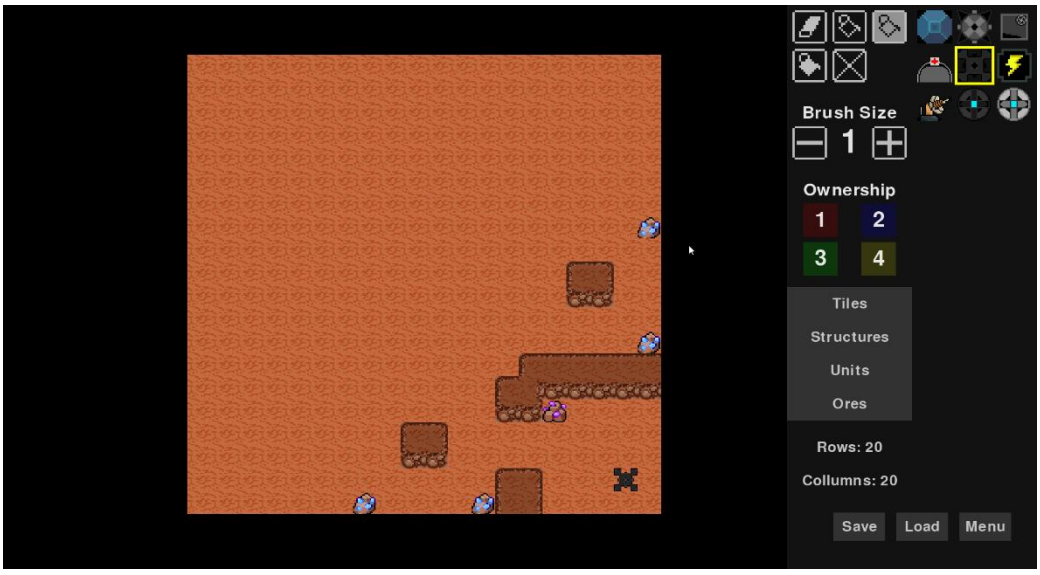
Proiectul are două părți importante:

-Editorul de hărți: Locul în care utilizatorul poate creea nivele.

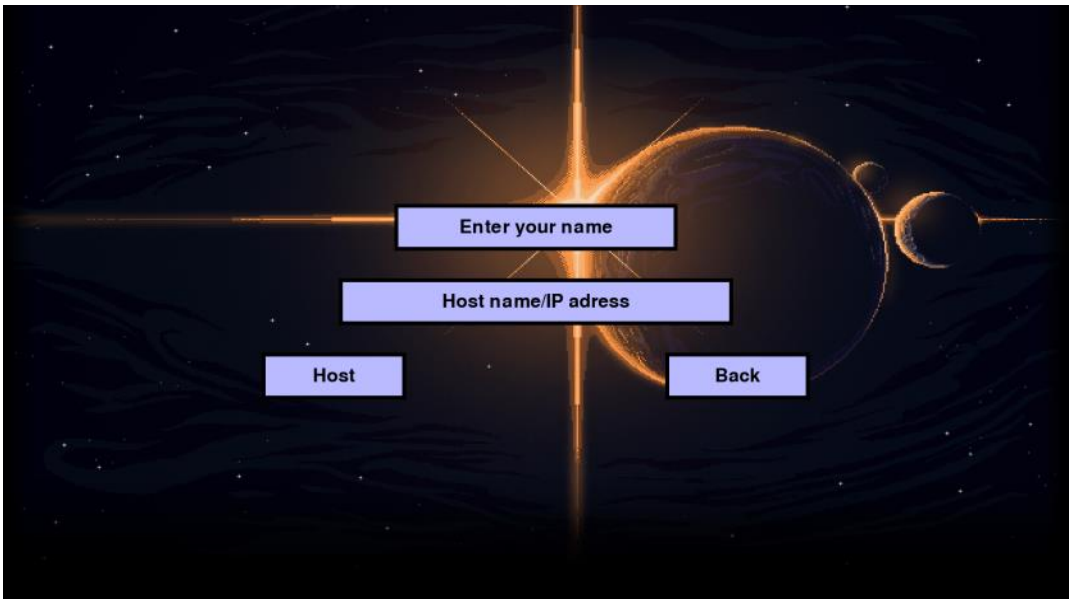
-Gameplay-ul: Locul în care se petrece acțiunea.

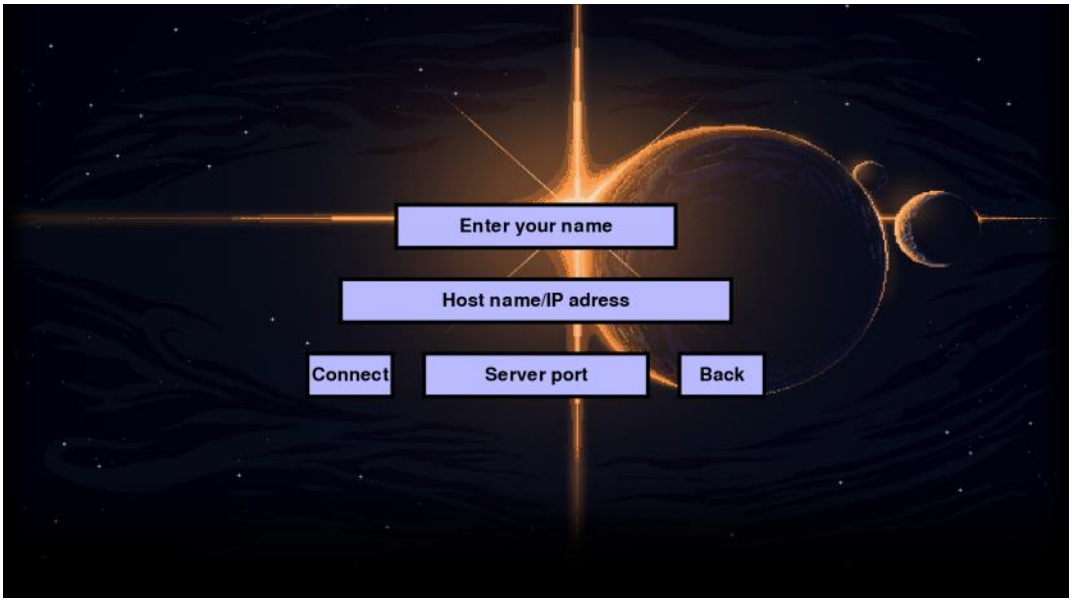
Pe apasarea butonului “Map Editor”, utilizatorul este prezentat cu diferite “unelte” de creație in dreapta ecranului.



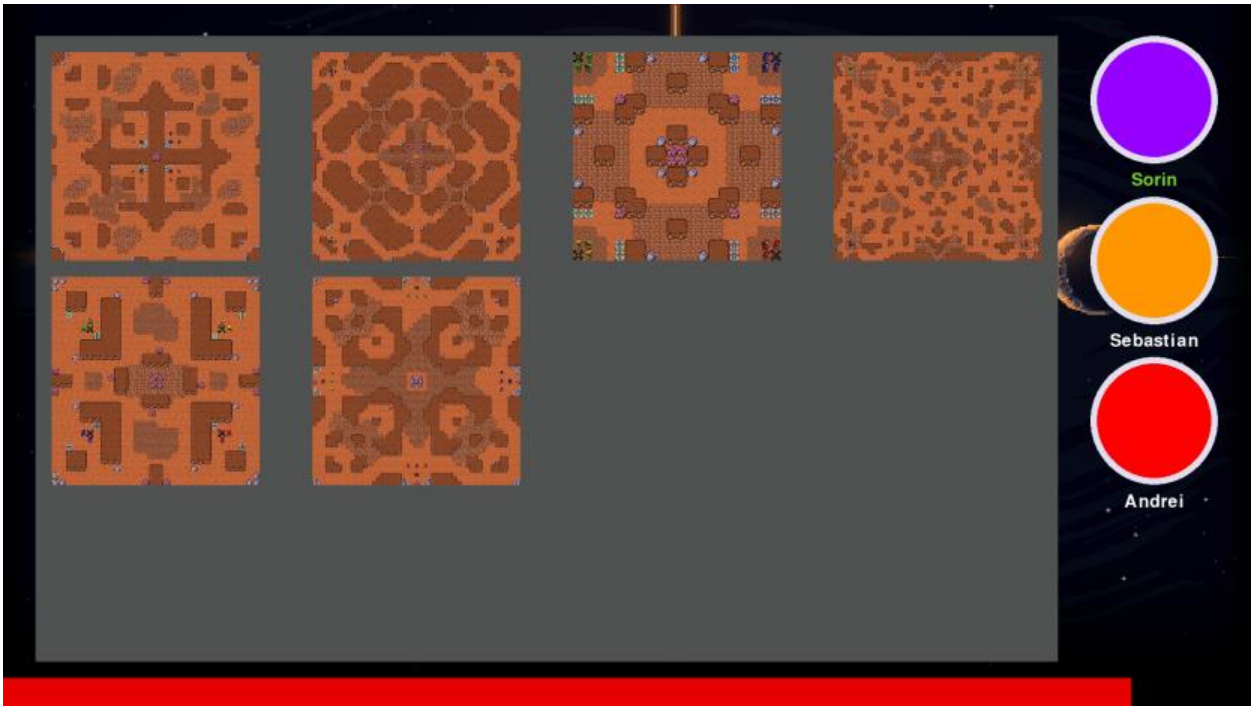


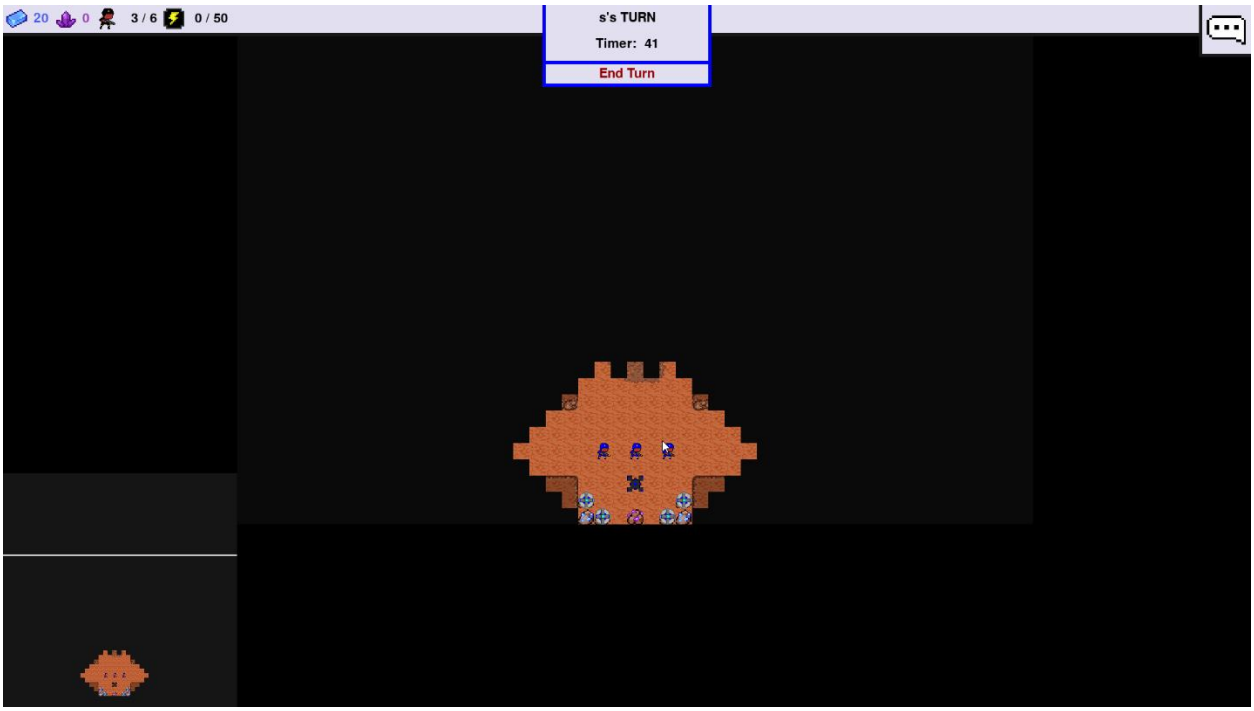
Utilizatorul are două opțiuni de a se juca cu alți utilizatori. Fie apasă pe butonul “Host”, care practic il face gazda meciului, fie apasă pe butonul “Join”, unde se va putea conecta la o altă gazdă.

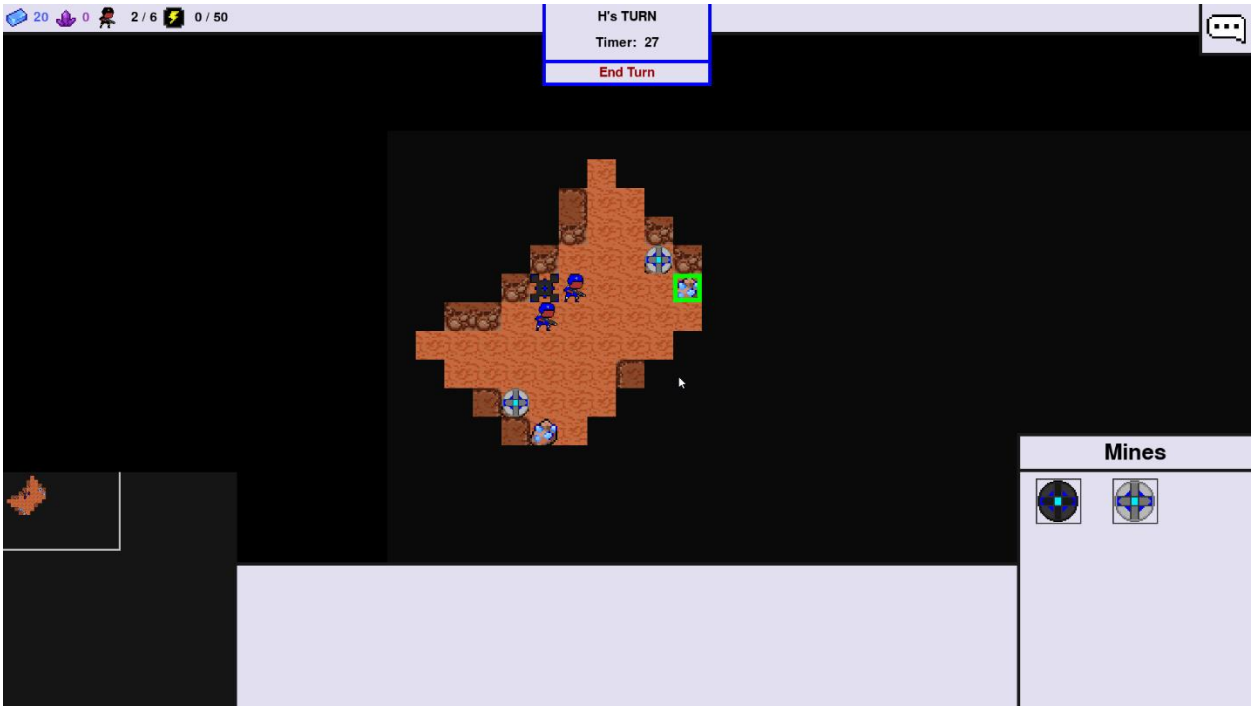





Port: 65432
Sorin
Sebastian
Andrei
Ready
Ready
Ready


Sorin
Sebastian
Andrei


20 0 3 / 6 0 / 50
s's TURN
Timer: 41
End Turn


20 0 2 / 6 0 / 50
H's TURN
Timer: 27
End Turn
Mines

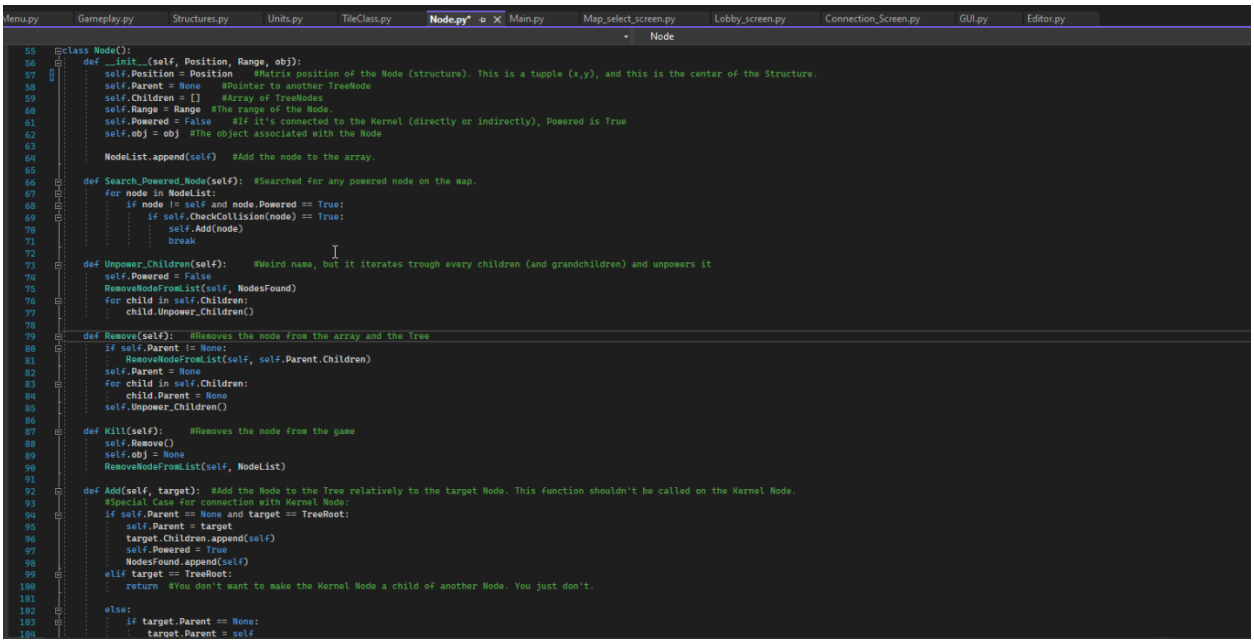
Modul de realizare

Proiectul a fost realizat folosind limbajul de programare Python 3, în decursul a șase luni, din octombrie 2022 si până în aprilie 2023. De asemenea, au mai fost folosite și programe de desenare 2D: Piskel și Aesprite.

Python este un limbaj de programare interpretat, în care codul nu este compilat la începutul rulării, cum funcționează C, C++, Go sau Haskel. Este un limbaj foarte flexibil, venind cu un installer de pachete de diferite module scrise de alți utilizatori. Un modul important folosit în proiect este pygame, care oferă funcții folosite pentru desenarea imaginilor.

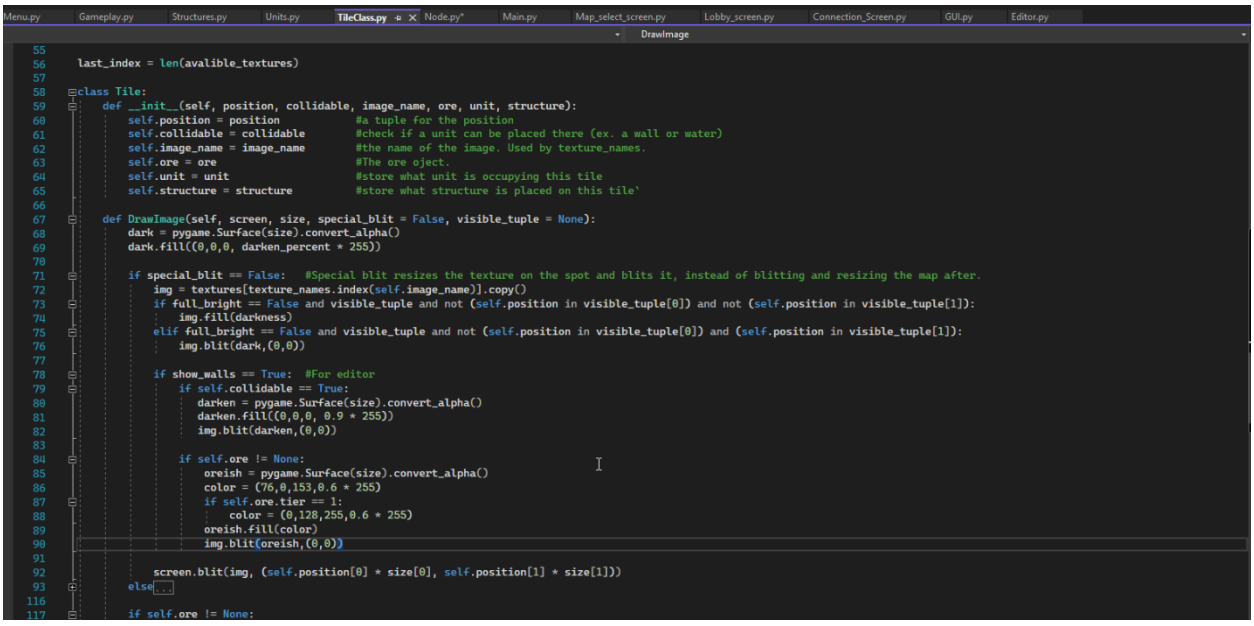
Cele mai importante părți ale programului sunt:

-Clasa de noduri:



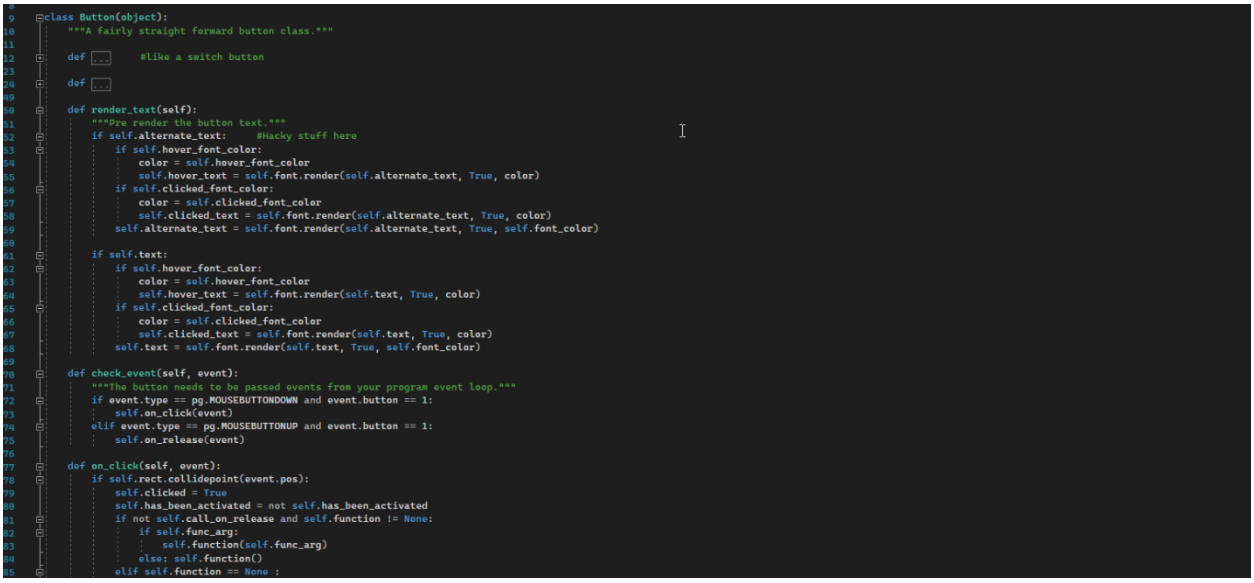
```python
class Node():
    def __init__(self, Position, Range, obj):
        self.Position = Position    #Matrix position of the Node (structure). This is a tupple (x,y), and this is the center of the Structure.
        self.Parent = None    #Pointer to another TreeNode
        self.Children = []    #Array of TreeNodes
        self.Range = Range  #The range of the Node.
        self.Powered = False    #If it's connected to the Kernel (directly or indirectly), Powered is True
        self.obj = obj  #The object associated with the Node

        NodeList.append(self)   #Add the node to the array.

    def Search_Powered_Node(self):  #Searched for any powered node on the map.
        for node in NodeList:
            if node != self and node.Powered == True:
                if self.CheckCollision(node) == True:
                    self.Add(node)
                    break

    def Unpower_Children(self):     #Weird name, but it iterates trough every children (and grandchildren) and unpowers it
        self.Powered = False
        RemoveNodeFromList(self, NodesFound)
        for child in self.Children:
            child.Unpower_Children()

    def Remove(self):   #Removes the node from the array and the Tree
        if self.Parent != None:
            RemoveNodeFromList(self, self.Parent.Children)
        self.Parent = None
        for child in self.Children:
            child.Parent = None
        self.Unpower_Children()

    def Kill(self):     #Removes the node from the game
        self.Remove()
        self.obj = None
        RemoveNodeFromList(self, NodeList)

    def Add(self, target):  #Add the Node to the Tree relatively to the target Node. This function shouldn't be called on the Kernel Node.
        #Special Case for connection with Kernel Node:
        if self.Parent == None and target == TreeRoot:
            self.Parent = target
            target.Children.append(self)
            self.Powered = True
            NodesFound.append(self)
        elif target == TreeRoot:
            return  #You don't want to make the Kernel Node a child of another Node. You just don't.

        else:
            if target.Parent == None:
                target.Parent = self
```

-Clasa de tile-uri:



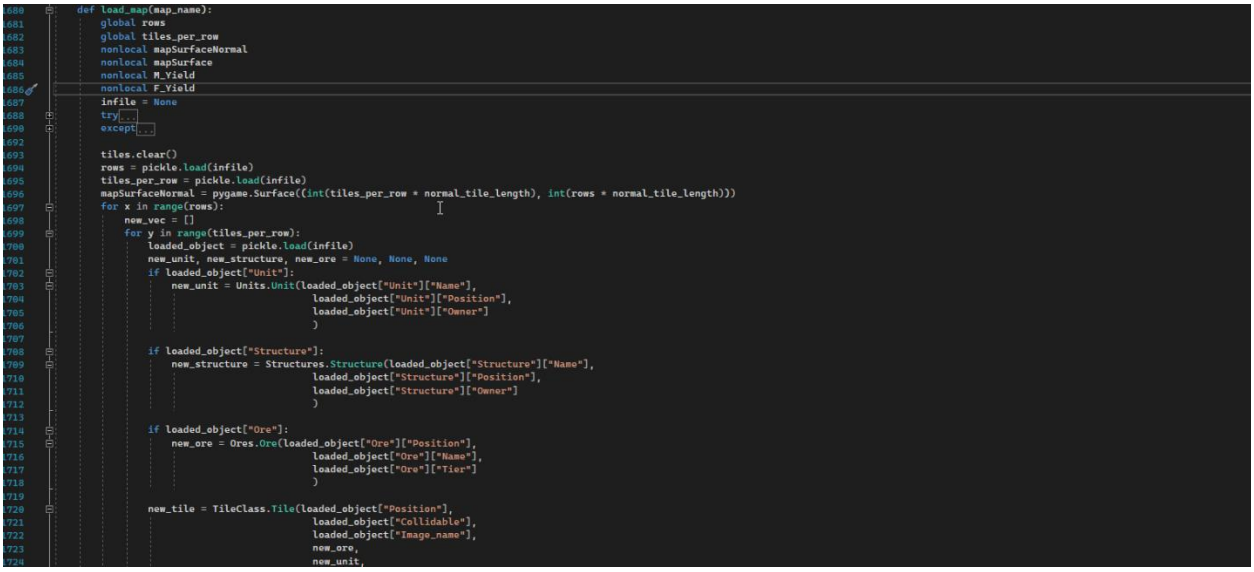
```python
last_index = len(avalible_textures)

class Tile:
    def __init__(self, position, collidable, image_name, ore, unit, structure):
        self.position = position            #a tuple for the position
        self.collidable = collidable        #check if a unit can be placed there (ex. a wall or water)
        self.image_name = image_name        #the name of the image. Used by texture_names.
        self.ore = ore                      #The ore oject.
        self.unit = unit                    #store what unit is occupying this tile
        self.structure = structure          #store what structure is placed on this tile'

    def DrawImage(self, screen, size, special_blit = False, visible_tuple = None):
        dark = pygame.Surface(size).convert_alpha()
        dark.fill((0,0,0, darken_percent * 255))

        if special_blit == False:   #Special blit resizes the texture on the spot and blits it, instead of blitting and resizing the map after.
            img = textures[texture_names.index(self.image_name)].copy()
            if full_bright == False and visible_tuple and not (self.position in visible_tuple[0]) and not (self.position in visible_tuple[1]):
                img.fill(darkness)
            elif full_bright == False and visible_tuple and not (self.position in visible_tuple[0]) and (self.position in visible_tuple[1]):
                img.blit(dark,(0,0))

            if show_walls == True:  #For editor
                if self.collidable == True:
                    darken = pygame.Surface(size).convert_alpha()
                    darken.fill((0,0,0, 0.9 * 255))
                    img.blit(darken,(0,0))

                if self.ore != None:
                    oreish = pygame.Surface(size).convert_alpha()
                    color = (76,0,153,0.6 * 255)
                    if self.ore.tier == 1:
                        color = (0,128,255,0.6 * 255)
                    oreish.fill(color)
                    img.blit(oreish,(0,0))

            screen.blit(img, (self.position[0] * size[0], self.position[1] * size[1]))
        else...

        if self.ore != None:
```

-Clasa de butoane:



```
class Button(object):
    """A fairly straight forward button class."""

    def ...     #Like a switch button

    def ...

    def render_text(self):
        """Pre render the button text."""
        if self.alternate_text:     #Hacky stuff here
            if self.hover_font_color:
                color = self.hover_font_color
                self.hover_text = self.font.render(self.alternate_text, True, color)
            if self.clicked_font_color:
                color = self.clicked_font_color
                self.clicked_text = self.font.render(self.alternate_text, True, color)
            self.alternate_text = self.font.render(self.alternate_text, True, self.font_color)

        if self.text:
            if self.hover_font_color:
                color = self.hover_font_color
                self.hover_text = self.font.render(self.text, True, color)
            if self.clicked_font_color:
                color = self.clicked_font_color
                self.clicked_text = self.font.render(self.text, True, color)
            self.text = self.font.render(self.text, True, self.font_color)

    def check_event(self, event):
        """The button needs to be passed events from your program event loop."""
        if event.type == pg.MOUSEBUTTONDOWN and event.button == 1:
            self.on_click(event)
        elif event.type == pg.MOUSEBUTTONUP and event.button == 1:
            self.on_release(event)

    def on_click(self, event):
        if self.rect.collidepoint(event.pos):
            self.clicked = True
            self.has_been_activated = not self.has_been_activated
            if not self.call_on_release and self.function != None:
                if self.func_arg:
                    self.function(self.func_arg)
                else: self.function()
            elif self.function == None :
```

-Funcția de încărcare a nivelelor:



```
    def load_map(map_name):
        global rows
        global tiles_per_row
        nonlocal mapSurfaceNormal
        nonlocal mapSurface
        nonlocal M_Yield
        nonlocal F_Yield
        infile = None
        try...
        except...

        tiles.clear()
        rows = pickle.load(infile)
        tiles_per_row = pickle.load(infile)
        mapSurfaceNormal = pygame.Surface((int(tiles_per_row * normal_tile_length), int(rows * normal_tile_length)))
        for x in range(rows):
            new_vec = []
            for y in range(tiles_per_row):
                loaded_object = pickle.load(infile)
                new_unit, new_structure, new_ore = None, None, None
                if loaded_object["Unit"]:
                    new_unit = Units.Unit(loaded_object["Unit"]["Name"],
                                          loaded_object["Unit"]["Position"],
                                          loaded_object["Unit"]["Owner"]
                                          )

                if loaded_object["Structure"]:
                    new_structure = Structures.Structure(loaded_object["Structure"]["Name"],
                                          loaded_object["Structure"]["Position"],
                                          loaded_object["Structure"]["Owner"]
                                          )

                if loaded_object["Ore"]:
                    new_ore = Ores.Ore(loaded_object["Ore"]["Position"],
                                          loaded_object["Ore"]["Name"],
                                          loaded_object["Ore"]["Tier"]
                                          )

                new_tile = TileClass.Tile(loaded_object["Position"],
                                          loaded_object["Collidable"],
                                          loaded_object["Image_name"],
                                          new_ore,
                                          new_unit,
```

Resursele necesare utilizarii

Pentru folosirea proiectului este necesar un dispozitiv ce poate rula sistemul de operare Windows 7, cerințele de system fiind incluse în cerintele sistemului de operare.

Pentru a putea crea meciuri ca “gazdă”, este necesar ca utilizatorul să fie familiar cu procedee de port-forwarding sau utilizarea de programe ajutătoare precum “ngrok”. (https://ngrok.com/download)

Bibliografie

Programe folosite pentru realizarea proiectului: