CS-223
WHITEBOX TESTING
DOCUMENT

for

Project 3
Virtual Tour Based Game

Prepared by:
Group-20
Inderpreet Singh Chera - 160101035
Shubhendu Patidar - 160101068
Siddharth Sharma - 160101071

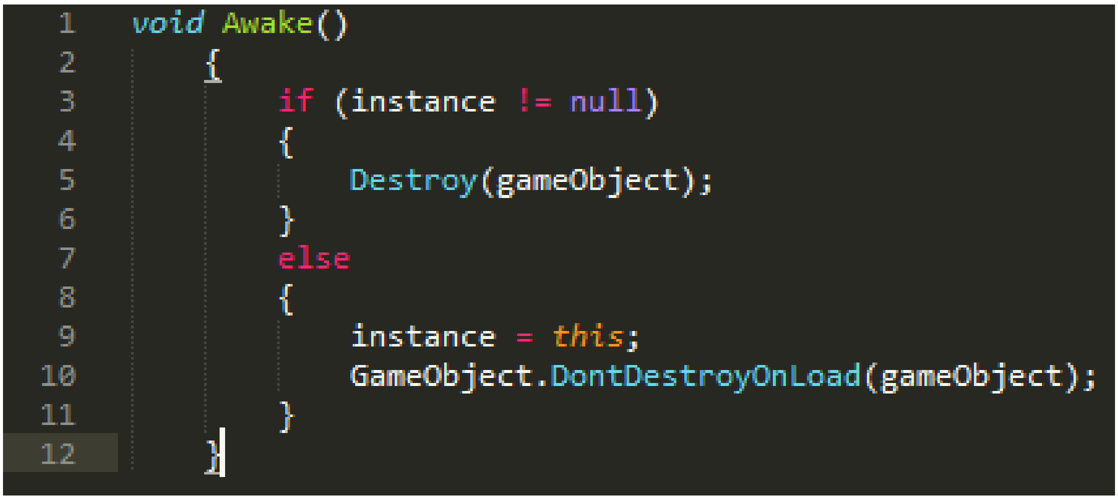
April 22, 2018

Contents

1 White Box Testing

1.1 Module: AudioScript

1.1.1 Funtion: Awake()



```
1   void Awake()
2       {
3           if (instance != null)
4           {
5               Destroy(gameObject);
6           }
7           else
8           {
9               instance = this;
10              GameObject.DontDestroyOnLoad(gameObject);
11          }
12      }
```

Figure 1.1: Code for Awake() function

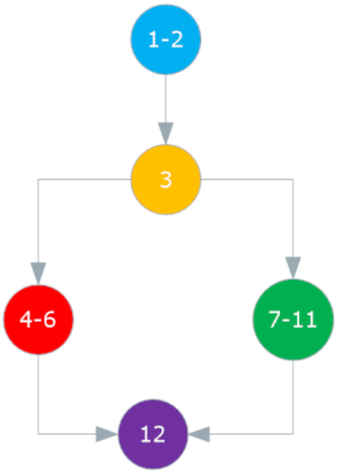


Figure 1.2: CFG for Awake() function

1.1.1.1 Calculation of Linearly Independent Paths

Number of Linearly independent paths = Number of Edges - Number of nodes + 2 = 2

1.1.1.2 Linearly Independent Paths

- (1-2)->3->(4-6)->12

  Testcase: instance != null (meaning to make sure that only one instance of script is active)

  Expected Output: destroy duplicate instances

  Observed Output: destroy duplicate instances

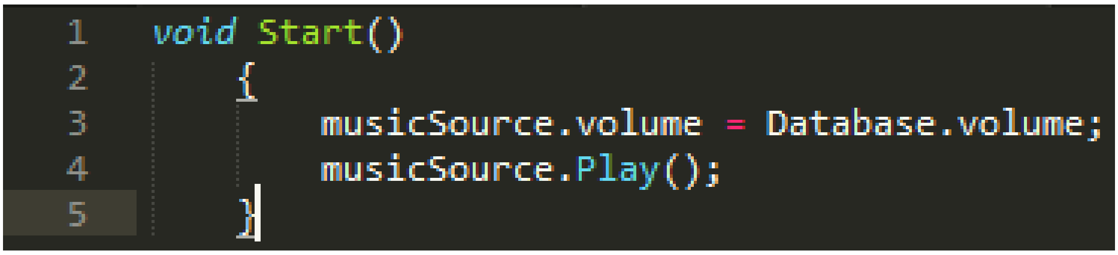
- (1-2)->3->(7-11)->12

  Testcase: instance = null

  Expected Output: assign this instance to the attached game object

  Observed Output: assign this instance to the attached game object

1.1.2 Funtion: Start()



```
void Start()
    {
        musicSource.volume = Database.volume;
        musicSource.Play();
    }
```

Figure 1.3: Code for Start() function

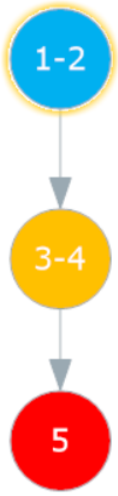


Figure 1.4: CFG for Start() function

1.1.2.1 Calculation of Linearly Independent Paths

Number of Linearly independent paths = Number of Edges - Number of nodes + 2 = 1

1.1.2.2 Linearly Independent Paths

- (1-2)->(3-4)->5

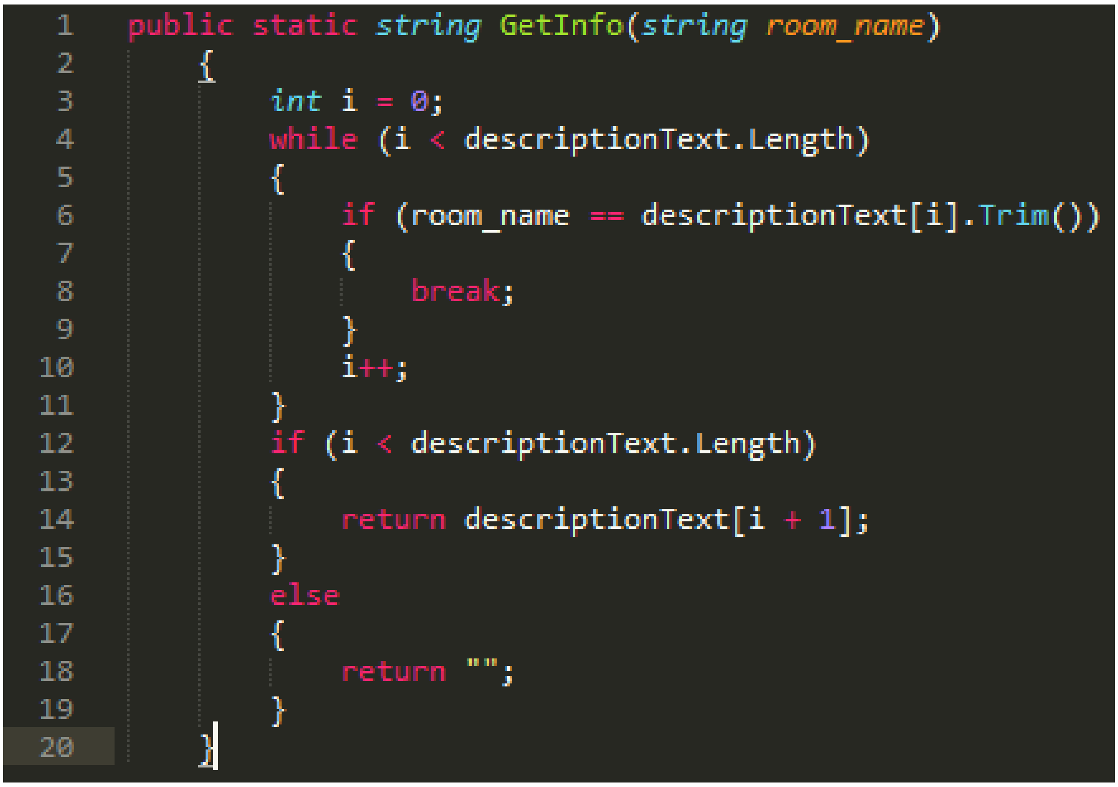
  Testcase: All possible cases

  Expected Output: get music volume from database and start playing music

  Observed Output: get music volume from database and start playing music

1.2 Module: Database

1.2.1 Function: GetInfo()



```
public static string GetInfo(string room_name)
    {
        int i = 0;
        while (i < descriptionText.Length)
        {
            if (room_name == descriptionText[i].Trim())
            {
                break;
            }
            i++;
        }
        if (i < descriptionText.Length)
        {
            return descriptionText[i + 1];
        }
        else
        {
            return "";
        }
    }
```

Figure 1.5: Code for GetInfo() function

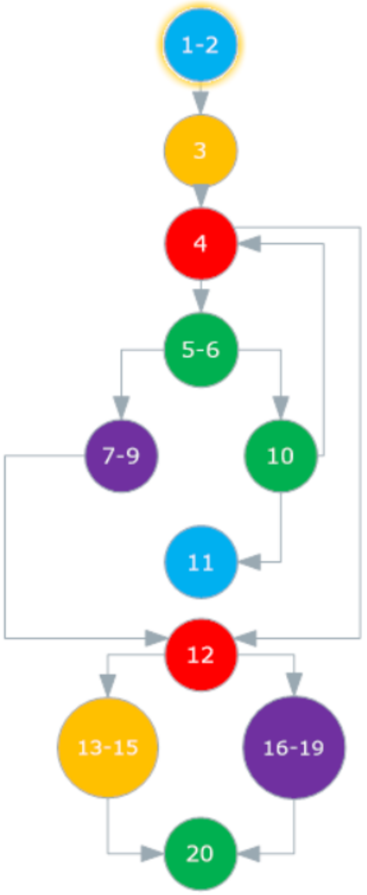


Figure 1.6: CFG for GetInfo() function

1.2.1.1 Calculation of Linearly Independent Paths

Number of Linearly independent paths = Number of Edges - Number of nodes + 2 = 4

1.2.1.2 Linearly Independent Paths

- (1-2)->3->4->(5-6)->(7-9)->12->(13-15)->20

  Testcase: descriptionText[0] == room_name

  Expected Output: Returns the Room Description a for particular room ”room_name”

  Observed Output: Returns the Room Description a for particular room ”room_name”

- (1-2)->3->4->(5-6)->(7-9)->12->(16-19)->20

Testcase: descriptionText[0] == room_name, and descriptionText.Length = 1

Expected Output: Returns the Room Description a for particular room ”room_name”

Observed Output: Returns empty string

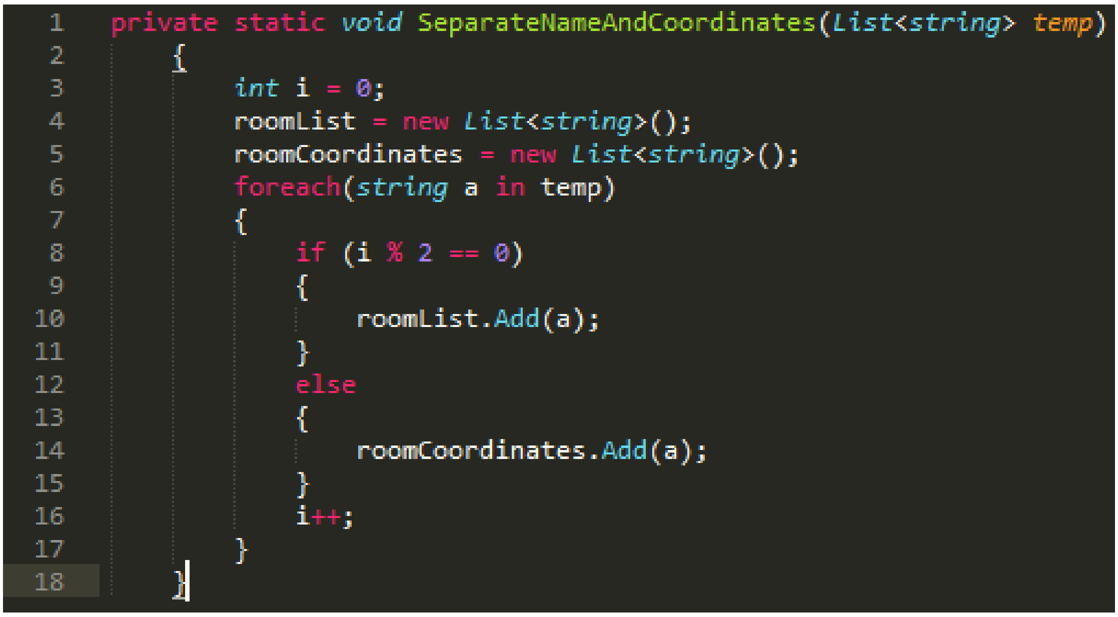
- (1-2)->3->4->(5-6)->10->4->(5-6)->10->11->12->(16-19)->20

  Testcase: descriptionText.Length = 1

  Expected Output: returns empty string

  Observed Output: Returns the Room Description a for particular room ”room_name”

1.2.2 Funtion: SeperateNameAndCoordinates()



```
private static void SeparateNameAndCoordinates(List<string> temp)
    {
        int i = 0;
        roomList = new List<string>();
        roomCoordinates = new List<string>();
        foreach(string a in temp)
        {
            if (i % 2 == 0)
            {
                roomList.Add(a);
            }
            else
            {
                roomCoordinates.Add(a);
            }
            i++;
        }
    }
```

Figure 1.7: Code for SeparateNameAndCoordinates() function

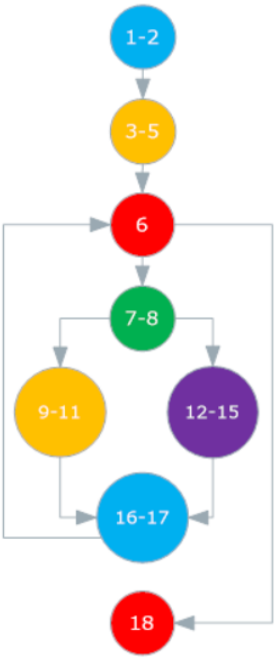


Figure 1.8: CFG for SeparateNameAndCoordinates() function

1.2.2.1 Calculation of Linearly Independent Paths

Number of Linearly independent paths = Number of Edges - Number of nodes + 2 = 3

1.2.2.2 Linearly Independent Paths

- (1-2)->(3-5)->6->(7-8)->(9-11)->(16-17)->6->18

  Testcase: temp.Count = 1

  Expected Output: adds room names and coordinates from temp to roomList and roomCoordinates respectively

  Observed Output: adds room names and coordinates from temp to roomList and roomCoordinates respectively

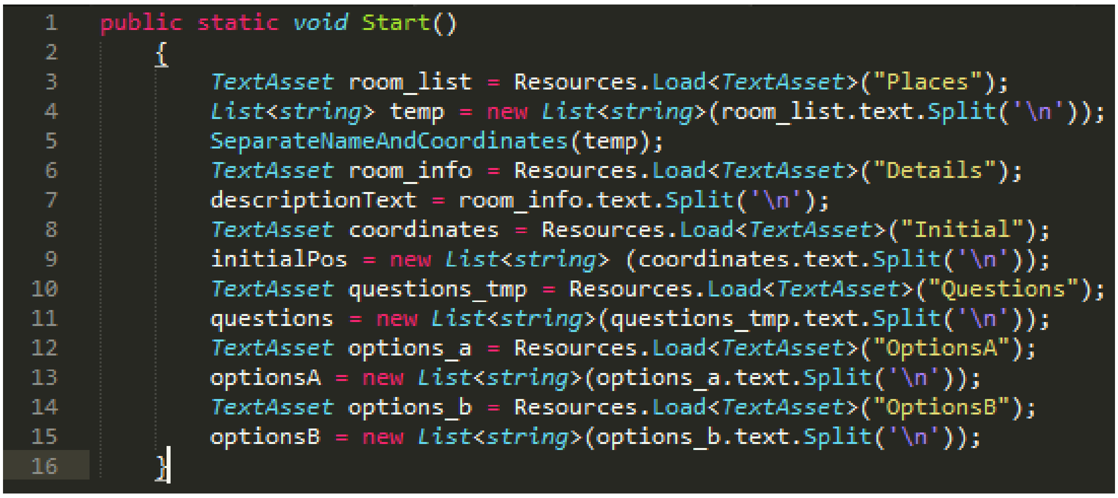
- (1-2)->(3-5)->6->(7-8)->(12-15)->(16-17)->6->18

  Testcase: temp.Count = 2

  Expected Output: adds room names and coordinates from temp to roomList and roomCoordinates respectively

  Observed Output: adds room names and coordinates from temp to roomList and roomCoordinates respectively

1.2.3 Funtion: Start()



```
1   public static void Start()
2       {
3           TextAsset room_list = Resources.Load<TextAsset>("Places");
4           List<string> temp = new List<string>(room_list.text.Split('\n'));
5           SeparateNameAndCoordinates(temp);
6           TextAsset room_info = Resources.Load<TextAsset>("Details");
7           descriptionText = room_info.text.Split('\n');
8           TextAsset coordinates = Resources.Load<TextAsset>("Initial");
9           initialPos = new List<string> (coordinates.text.Split('\n'));
10          TextAsset questions_tmp = Resources.Load<TextAsset>("Questions");
11          questions = new List<string>(questions_tmp.text.Split('\n'));
12          TextAsset options_a = Resources.Load<TextAsset>("OptionsA");
13          optionsA = new List<string>(options_a.text.Split('\n'));
14          TextAsset options_b = Resources.Load<TextAsset>("OptionsB");
15          optionsB = new List<string>(options_b.text.Split('\n'));
16      }
```

Figure 1.9: Code for Start() function

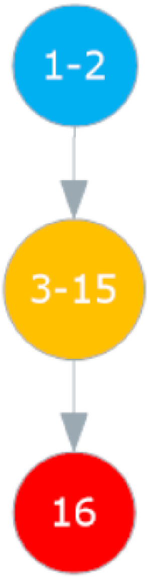


Figure 1.10: CFG for Start() function

1.2.3.1 Calculation of Linearly Independent Paths

Number of Linearly independent paths = Number of Edges - Number of nodes + 2 = 1

1.2.3.2 Linearly Independent Paths

- (1-2)->(3-15)->16

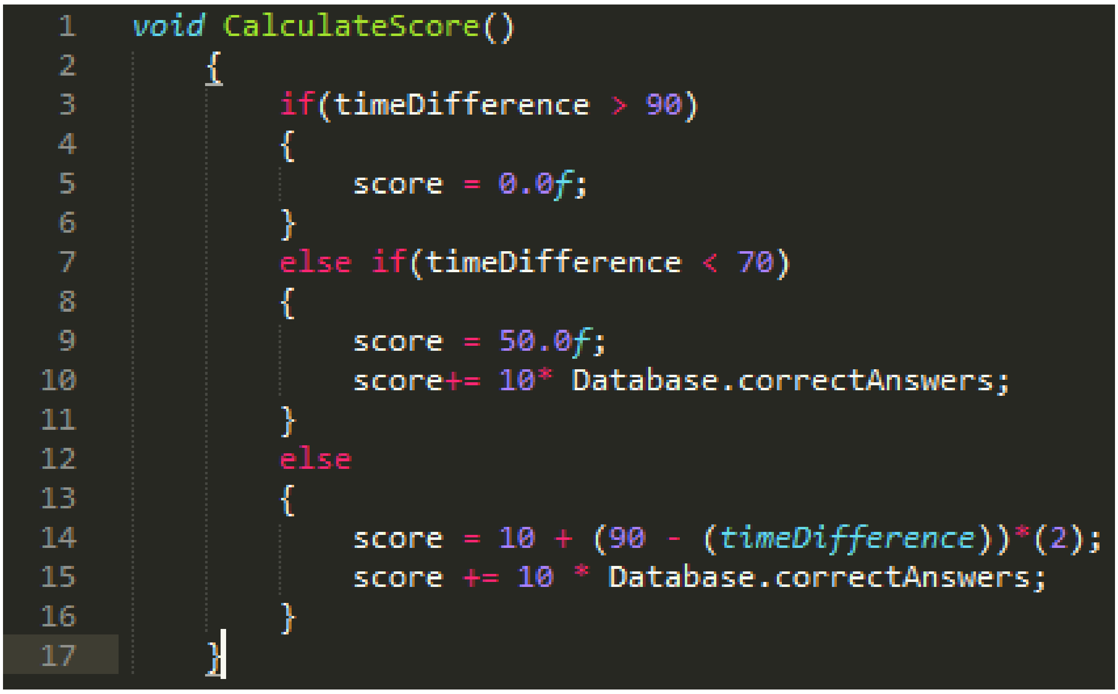
  Testcase: all possible cases

  Expected Output: get list of questions, optionsA, optionsB, room names, room details, and initial positions

  Observed Output: get list of questions, optionsA, optionsB, room names, room details, and initial positions

1.3 Module: DisplayScore

1.3.1 Funtion: CalculateScore()



```
void CalculateScore()
    {
        if(timeDifference > 90)
        {
            score = 0.0f;
        }
        else if(timeDifference < 70)
        {
            score = 50.0f;
            score+= 10* Database.correctAnswers;
        }
        else
        {
            score = 10 + (90 - (timeDifference))*(2);
            score += 10 * Database.correctAnswers;
        }
    }
```

Figure 1.11: Code for CalculateScore() function

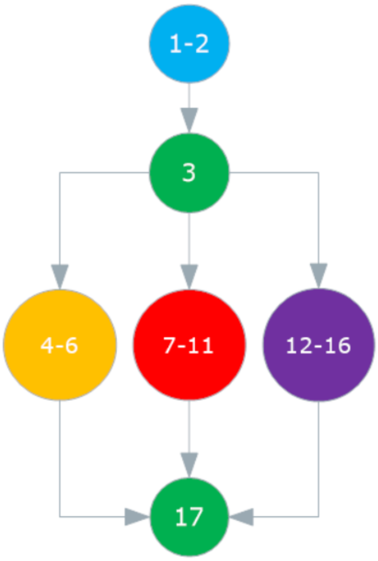


Figure 1.12: CFG for CalculateScore() function

1.3.1.1 Calculation of Linearly Independent Paths

Number of Linearly independent paths = Number of Edges - Number of nodes + 2 = 3

1.3.1.2 Linearly Independent Paths

- (1-2)->3->(4-6)->17

  Testcase: timeDifference > 90

  Expected Output: set score to 0

  Observed Output: set score to 0

- (1-2)->3->(7-11)->17

  Testcase: timeDifference < 70

  Expected Output: set score to 50 + 10 * correct answers

  Observed Output: set score to 50 + 10 * correct answers

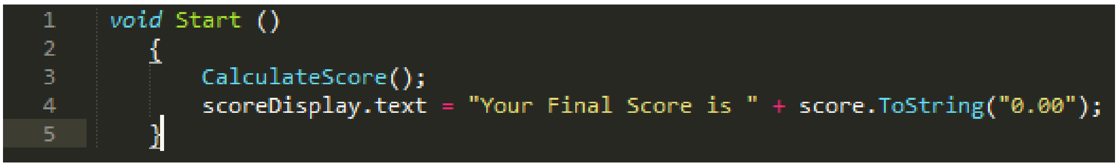
- (1-2)->3->(12-16)->17

  Testcase: 70 < timeDifference < 90

  Expected Output: set score to 60 - timeDifference + 10 * correct answers

  Observed Output: set score to 60 - timeDifference + 10 * correct answers

1.3.2 Funtion: Start()



```
1  void Start ()
2      {
3          CalculateScore();
4          scoreDisplay.text = "Your Final Score is " + score.ToString("0.00");
5      }
```

Figure 1.13: Code for Start() function

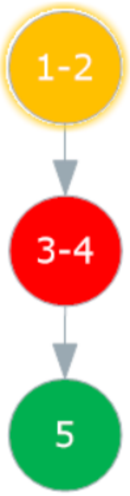


Figure 1.14: CFG for Start() function

1.3.2.1 Calculation of Linearly Independent Paths

Number of Linearly independent paths = Number of Edges - Number of nodes + 2 = 1

1.3.2.2 Linearly Independent Paths

- (1-2)->(3-4)->5

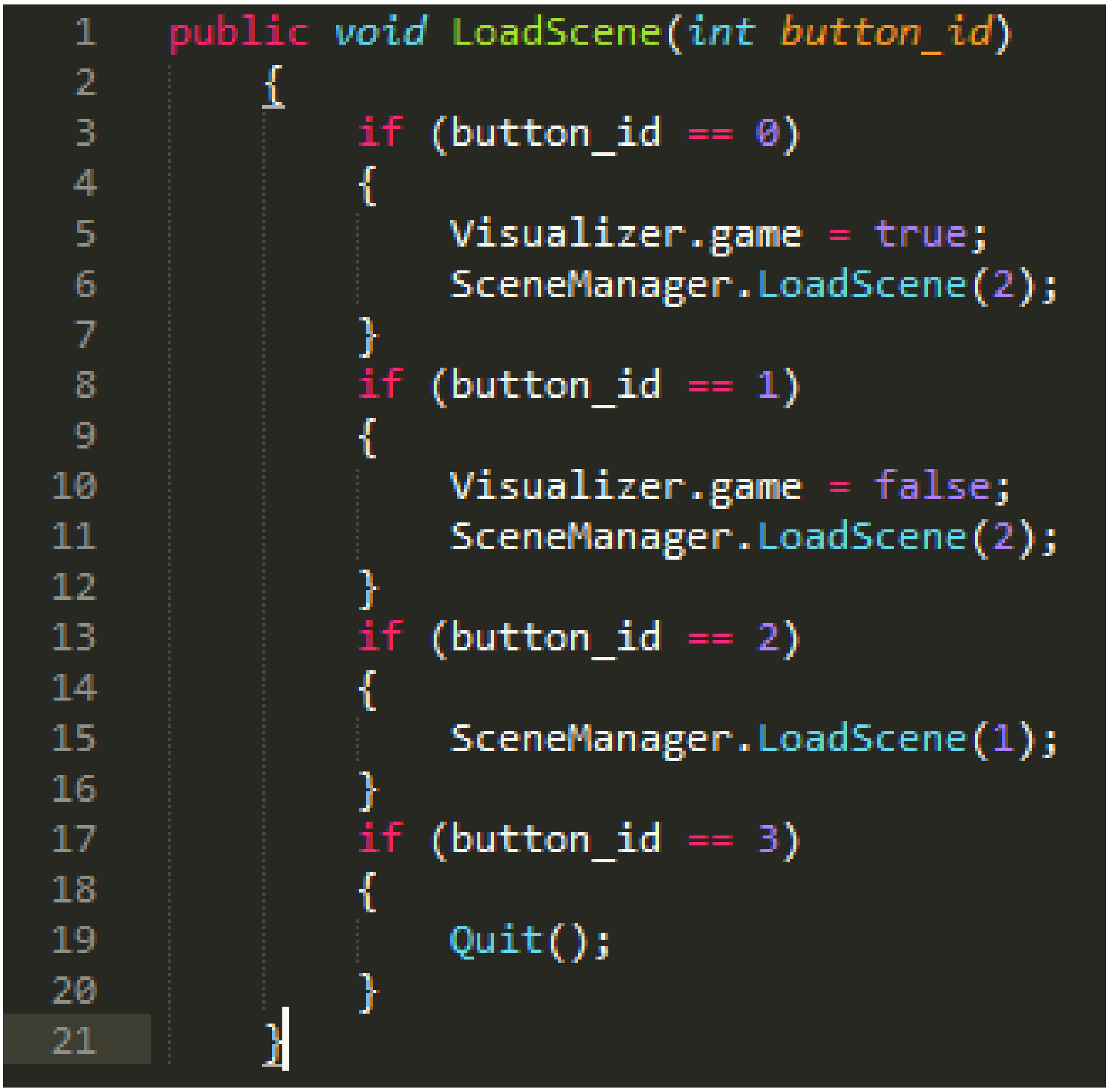
  Testcase: All possible cases

  Expected Output: calculate score and display it on the screen

  Observed Output: calculate score and display it on the screen

1.4 Module: MainmenuScripts

1.4.1 Funtion: LoadScene()



```
public void LoadScene(int button_id)
    {
        if (button_id == 0)
        {
            Visualizer.game = true;
            SceneManager.LoadScene(2);
        }
        if (button_id == 1)
        {
            Visualizer.game = false;
            SceneManager.LoadScene(2);
        }
        if (button_id == 2)
        {
            SceneManager.LoadScene(1);
        }
        if (button_id == 3)
        {
            Quit();
        }
    }
```

Figure 1.15: Code for LoadScene() function

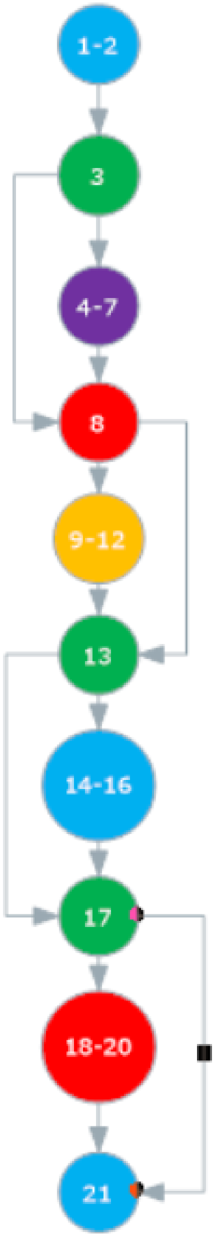


Figure 1.16: CFG for LoadScene() function

1.4.1.1 Calculation of Linearly Independent Paths

Number of Linearly independent paths = Number of Edges - Number of nodes + 2 = 5

1.4.1.2 Linearly Independent Paths

- (1-2)->3->(4-7)->8->13->17->21

  Testcase: button_id == 0

  Expected Output: Load new game

  Observed Output: Load new game

- (1-2)->3->8->(9-12)->13->17->21

  Testcase: button_id == 1

  Expected Output: Load training mode

  Observed Output: Load training mode

- (1-2)->3->8->13->(14-16)->17->21

  Testcase: button_id == 2

  Expected Output: load Settings scene

  Observed Output: load Settings scene

- (1-2)->3->8->13->17->(18-20)->21

  Testcase: button_id == 3

  Expected Output: exit application

  Observed Output: exit application

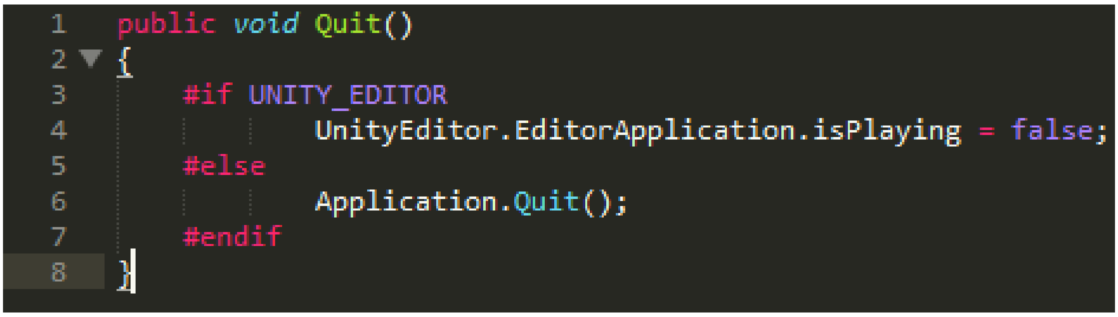
- (1-2)->3->8->13->17->21

  Testcase: validate = False

  Expected Output: Call onSignupFailed() and thus display ”signup failed” message.

  Observed Output: Displays ”signup failed” message.

1.4.2 Funtion: Quit()



```
public void Quit()
{
    #if UNITY_EDITOR
        UnityEditor.EditorApplication.isPlaying = false;
    #else
        Application.Quit();
    #endif
}
```

Figure 1.17: Code for Quit() function

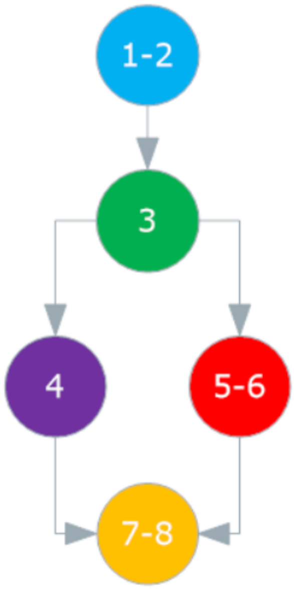


Figure 1.18: CFG for Quit() function

1.4.2.1 Calculation of Linearly Independent Paths

Number of Linearly independent paths = Number of Edges - Number of nodes + 2 = 2

1.4.2.2 Linearly Independent Paths

- (1-2)->3->4->(7-8)

  Testcase: UNITY_EDITOR = True (Game is being played in unity Game Engine)

  Expected Output: Exit back to Unity Editor

  Observed Output: Exit back to Unity Editor

- (1-2)->3->(5-6)->(7-8)

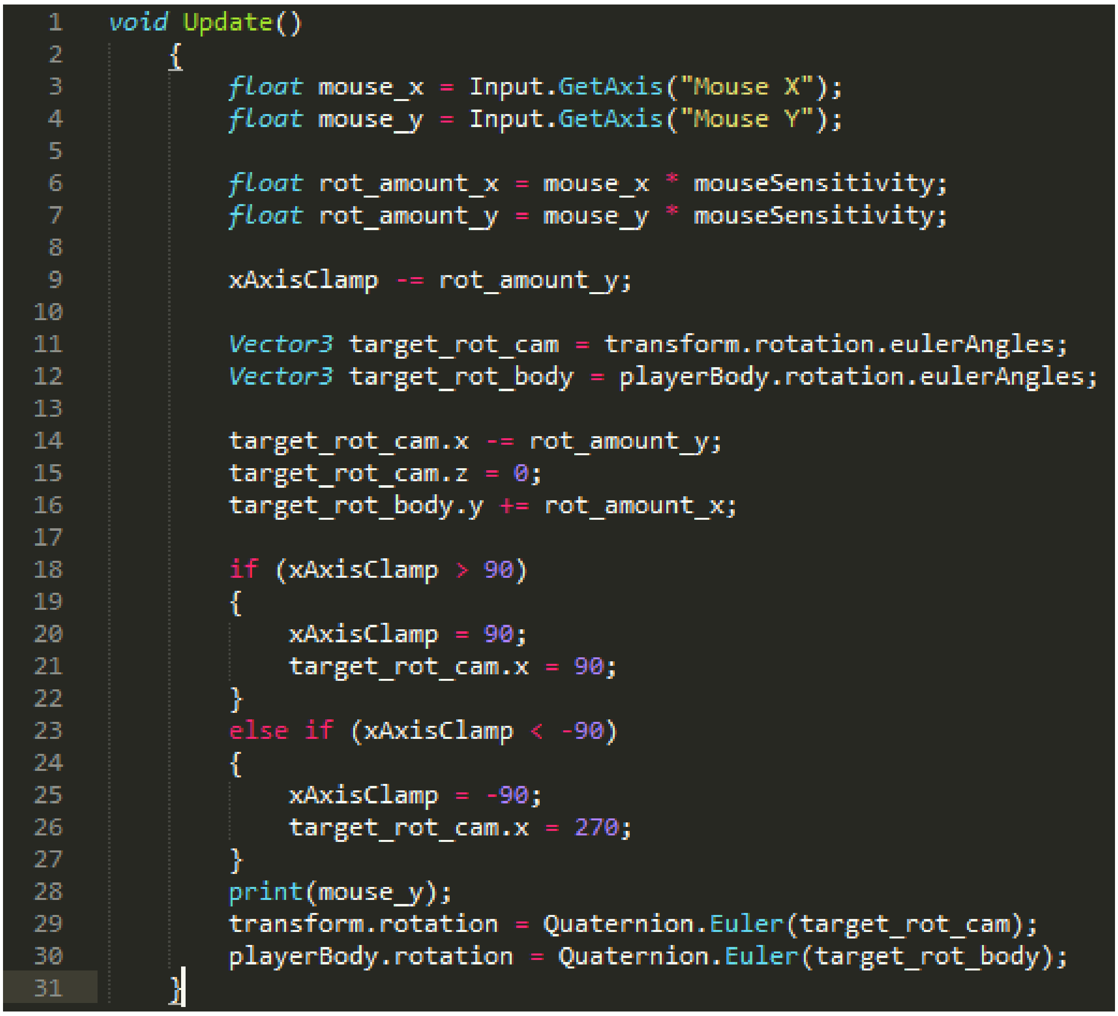
  Testcase: UNITY_EDITOR = False

  Expected Output: Exit Application

  Observed Output: Exit Application

1.5 Module: PlayerLook

1.5.1 Funtion: Update()



```
void Update()
    {
        float mouse_x = Input.GetAxis("Mouse X");
        float mouse_y = Input.GetAxis("Mouse Y");

        float rot_amount_x = mouse_x * mouseSensitivity;
        float rot_amount_y = mouse_y * mouseSensitivity;

        xAxisClamp -= rot_amount_y;

        Vector3 target_rot_cam = transform.rotation.eulerAngles;
        Vector3 target_rot_body = playerBody.rotation.eulerAngles;

        target_rot_cam.x -= rot_amount_y;
        target_rot_cam.z = 0;
        target_rot_body.y += rot_amount_x;

        if (xAxisClamp > 90)
        {
            xAxisClamp = 90;
            target_rot_cam.x = 90;
        }
        else if (xAxisClamp < -90)
        {
            xAxisClamp = -90;
            target_rot_cam.x = 270;
        }
        print(mouse_y);
        transform.rotation = Quaternion.Euler(target_rot_cam);
        playerBody.rotation = Quaternion.Euler(target_rot_body);
    }
```

Figure 1.19: Code for Update() function

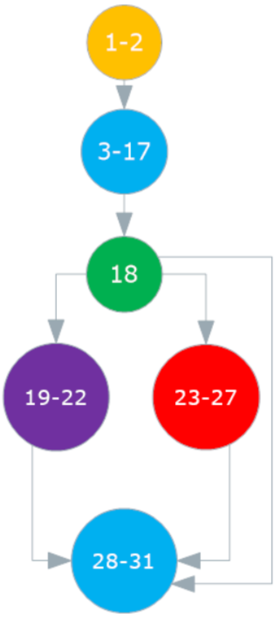


Figure 1.20: CFG for Update() function

1.5.1.1 Calculation of Linearly Independent Paths

Number of Linearly independent paths = Number of Edges - Number of nodes + 2 = 3

1.5.1.2 Linearly Independent Paths

- (1-2)->(3-17)->18->(19-22)->(28-31)

  Testcase: xAxisClamp > 90

  Expected Output: updates camera orientation

  Observed Output: updates camera orientation

- (1-2)->(3-17)->18->(23-27)->(28-31)

  Testcase: xAxisClamp < -90

  Expected Output: updates camera orientation

  Observed Output: updates camera orientation

- (1-2)->(3-17)->18->(28-31)

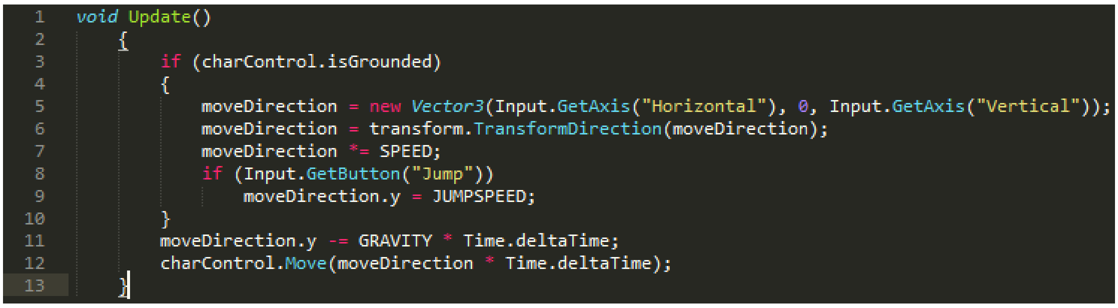
  Testcase: -90 <= xAxisClamp <= 90

  Expected Output: updates camera orientation

  Observed Output: updates camera orientation

1.6 Module: PlayerMove

1.6.1 Funtion: Update()



```
1   void Update()
2       {
3           if (charControl.isGrounded)
4           {
5               moveDirection = new Vector3(Input.GetAxis("Horizontal"), 0, Input.GetAxis("Vertical"));
6               moveDirection = transform.TransformDirection(moveDirection);
7               moveDirection *= SPEED;
8               if (Input.GetButton("Jump"))
9                   moveDirection.y = JUMPSPEED;
10          }
11          moveDirection.y -= GRAVITY * Time.deltaTime;
12          charControl.Move(moveDirection * Time.deltaTime);
13      }
```

Figure 1.21: Code for Update() function

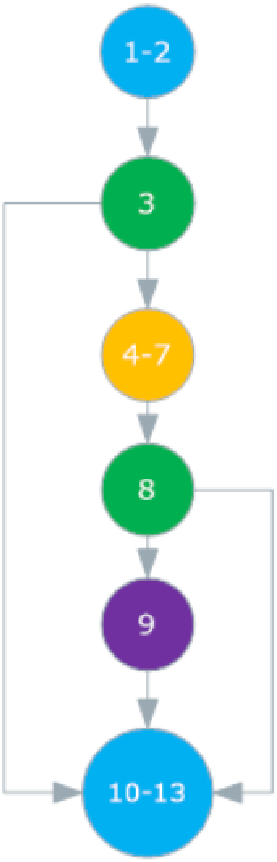


Figure 1.22: CFG for Update() function

1.6.1.1 Calculation of Linearly Independent Paths

Number of Linearly independent paths = Number of Edges - Number of nodes + 2 = 3

1.6.1.2 Linearly Independent Paths

- (1-2)->3->(4-7)->8->9->(10-13)

  Testcase: charControl.isGrounded = True, Input.GetButton = ”Jump”

  Expected Output: Player jumps in the direction of movement.

  Observed Output: Player jumps in the direction of movement.

- (1-2)->3->(10-13)

  Testcase: charControl.isGrounded = False

  Expected Output: Player falls with speed = GRAVITY

  Observed Output: Player falls with speed = GRAVITY

- (1-2)->3->(4-7)->8->(10-13)

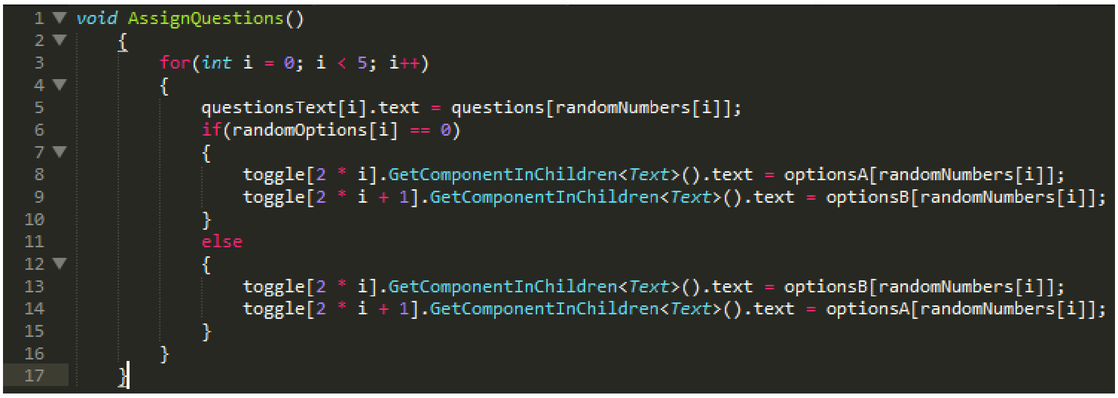
  Testcase: charControl.isGrounded = True, Input.GetButton != ”Jump”

  Expected Output: Player moves horizontally with speed = SPEED

  Observed Output: Player moves horizontally with speed = SPEED

1.7 Module: QuestionAnswers

1.7.1 Funtion: AssignQuestions()



```
void AssignQuestions()
    {
        for(int i = 0; i < 5; i++)
        {
            questionsText[i].text = questions[randomNumbers[i]];
            if(randomOptions[i] == 0)
            {
                toggle[2 * i].GetComponentInChildren<Text>().text = optionsA[randomNumbers[i]];
                toggle[2 * i + 1].GetComponentInChildren<Text>().text = optionsB[randomNumbers[i]];
            }
            else
            {
                toggle[2 * i].GetComponentInChildren<Text>().text = optionsB[randomNumbers[i]];
                toggle[2 * i + 1].GetComponentInChildren<Text>().text = optionsA[randomNumbers[i]];
            }
        }
    }
```

Figure 1.23: Code for AssignQuestions() function

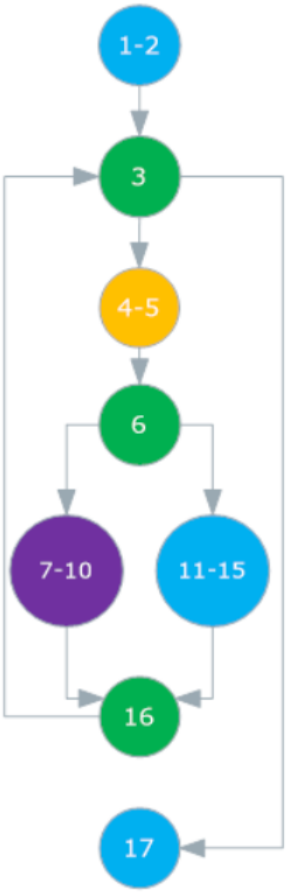


Figure 1.24: CFG for AssignQuestions() function

1.7.1.1 Calculation of Linearly Independent Paths

Number of Linearly independent paths = Number of Edges - Number of nodes + 2 = 3

1.7.1.2 Linearly Independent Paths

- (1-2)->3->(4-5)->6->(7-10)->16->3->(4-5)->6->(7-10)->16->3->(4-5)->6->(7-10)->16->3->(4-5)->6->(7-10)->16->3->(4-5)->6->(7-10)->16->17

  Testcase

  Expected Output: Set optionsA and optionsB list

  Observed Output: Set optionsA and optionsB list

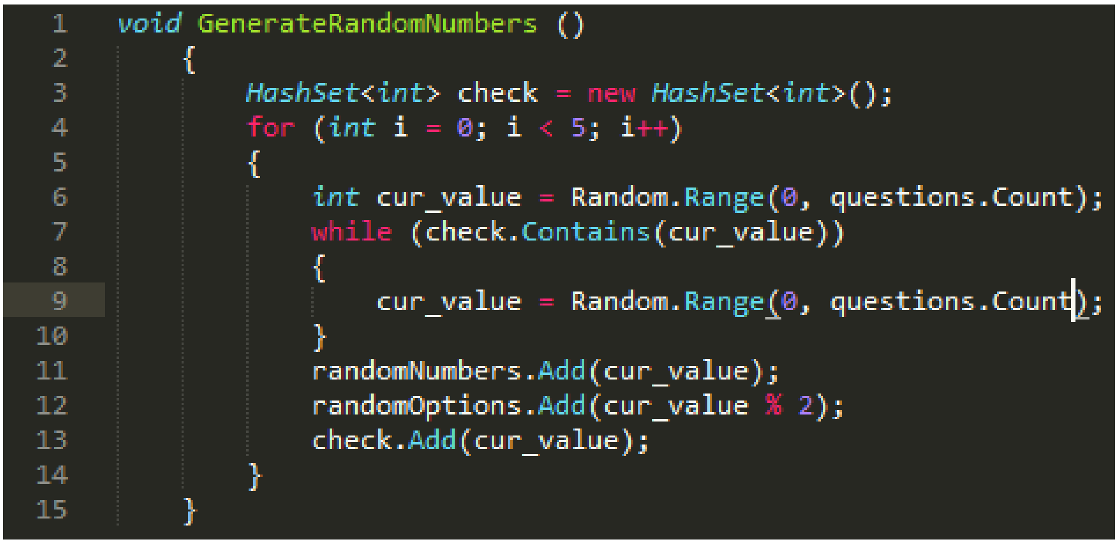
- (1-2)->3->(4-5)->6->(11-15)->16->3->17

  Testcase

  Expected Output: Set optionsA and optionsB list

Observed Output: Set optionsA and optionsB list

1.7.2 Funtion: GenerateRandomNumbers()



```
void GenerateRandomNumbers ()
    {
        HashSet<int> check = new HashSet<int>();
        for (int i = 0; i < 5; i++)
        {
            int cur_value = Random.Range(0, questions.Count);
            while (check.Contains(cur_value))
            {
                cur_value = Random.Range(0, questions.Count);
            }
            randomNumbers.Add(cur_value);
            randomOptions.Add(cur_value % 2);
            check.Add(cur_value);
        }
    }
```

Figure 1.25: Code for GenerateRandomNumbers() function

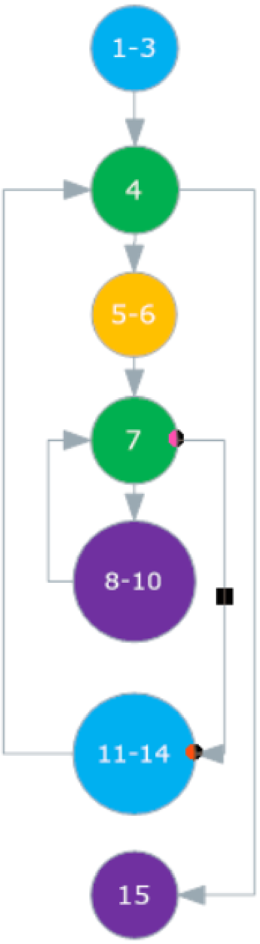


Figure 1.26: CFG for GenerateRandomNumbers() function

1.7.2.1 Calculation of Linearly Independent Paths

Number of Linearly independent paths = Number of Edges - Number of nodes + 2 = 3

1.7.2.2 Linearly Independent Paths

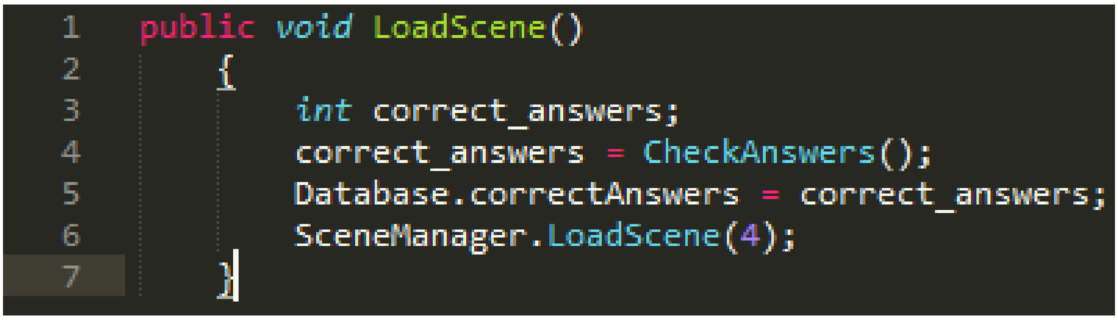
- (1-3)->4->(5-6)->7->(11-14)->4->(5-6)->7->(11-14)->4->(5-6)->7->(11-14)->4->(5-6)->7->(11-14)->4->(5-6)->7->(11-14)->4->15

  Testcase: Random.range(0, questions.Count) on line 6, always generates a new value that was not already in the hash table

  Expected Output: Populates the randomNumbers and randomOptions lists with 5 random number in range [0,questions.Count-1] (both inclusive), 0,1 respectively

Observed Output: Populates the randomNumbers and randomOptions lists with 5 random number in range [0,questions.Count-1] (both inclusive), 0,1 respectively

1.7.3 Funtion: LoadScene()



```
1   public void LoadScene()
2       {
3           int correct_answers;
4           correct_answers = CheckAnswers();
5           Database.correctAnswers = correct_answers;
6           SceneManager.LoadScene(4);
7       }
```

Figure 1.27: Code for LoadScene() function

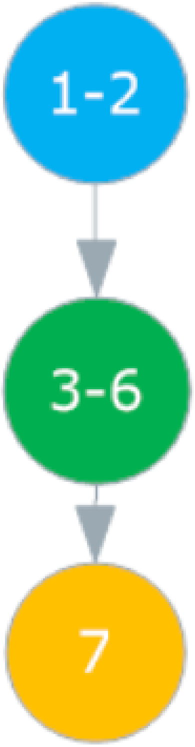


Figure 1.28: CFG for LoadScene() function

1.7.3.1 Calculation of Linearly Independent Paths

Number of Linearly independent paths = Number of Edges - Number of nodes + 2 = 1

1.7.3.2 Linearly Independent Paths

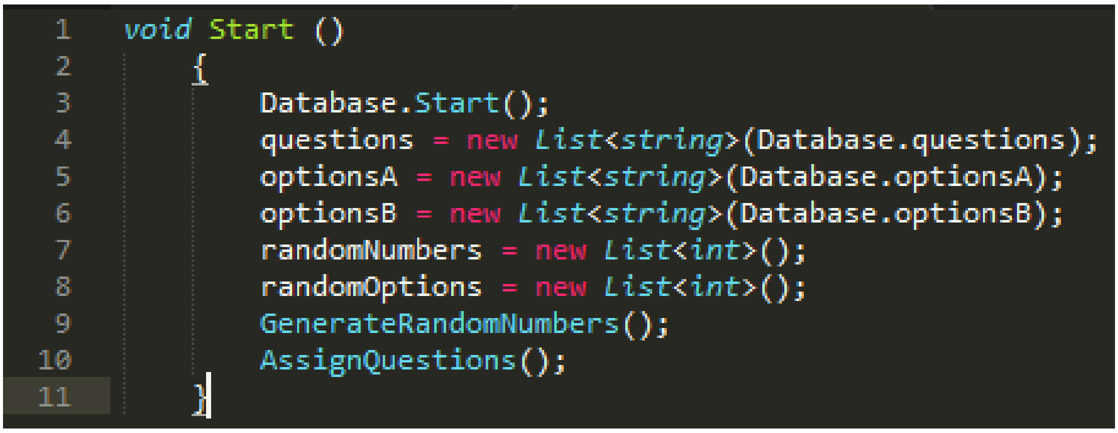
- (1-2)->(3-6)->7

  Testcase: All possible paths

  Expected Output: calls CheckAnswers() and sets the value of correct_answers and loads DisplayScore scene.

  Observed Output: calls CheckAnswers() and sets the value of correct_answers and loads DisplayScore scene.

1.7.4 Funtion: Start()



```
void Start ()
    {
        Database.Start();
        questions = new List<string>(Database.questions);
        optionsA = new List<string>(Database.optionsA);
        optionsB = new List<string>(Database.optionsB);
        randomNumbers = new List<int>();
        randomOptions = new List<int>();
        GenerateRandomNumbers();
        AssignQuestions();
    }
```

Figure 1.29: Code for Start() function

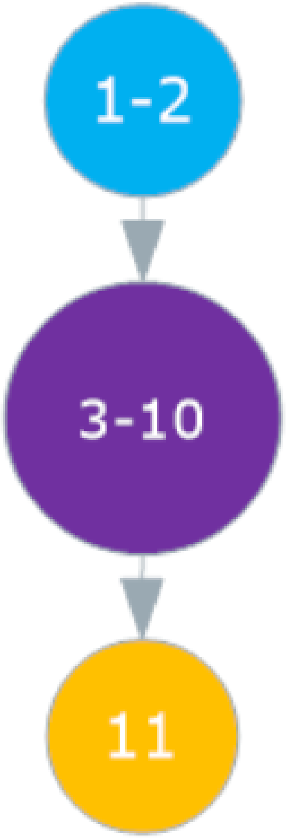


Figure 1.30: CFG for Start() function

1.7.4.1 Calculation of Linearly Independent Paths

Number of Linearly independent paths = Number of Edges - Number of nodes + 2 = 1

1.7.4.2 Linearly Independent Paths

- (1-2)->(3-10)->11

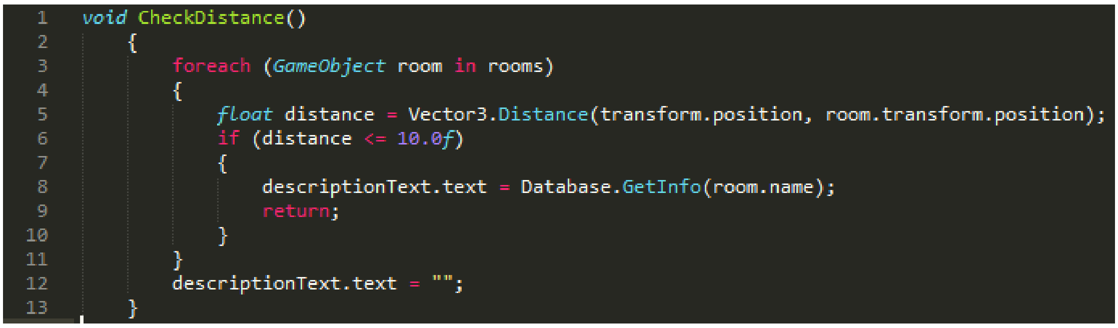
  Testcase: All possible cases

  Expected Output: populate database and set questions to ask

  Observed Output: populate database and set questions to ask

1.8 Module: Visualizer

1.8.1 Funtion: CheckDistance()



```
void CheckDistance()
    {
        foreach (GameObject room in rooms)
        {
            float distance = Vector3.Distance(transform.position, room.transform.position);
            if (distance <= 10.0f)
            {
                descriptionText.text = Database.GetInfo(room.name);
                return;
            }
        }
        descriptionText.text = "";
    }
```

Figure 1.31: Code for CheckDistance() function

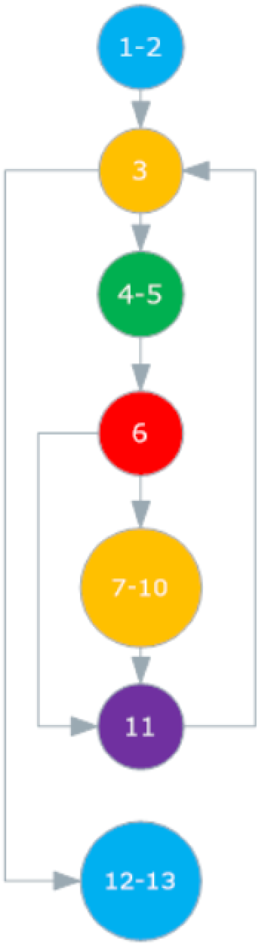


Figure 1.32: CFG for CheckDistance() function

1.8.1.1 Calculation of Linearly Independent Paths

Number of Linearly independent paths = Number of Edges - Number of nodes + 2 = 3

1.8.1.2 Linearly Independent Paths

- (1-2)->3->(4-5)->6->(7-9)

  Testcase: rooms.Count = 1, distance <= 10

  Expected Output: return description of nearest room

  Observed Output: return description of nearest room

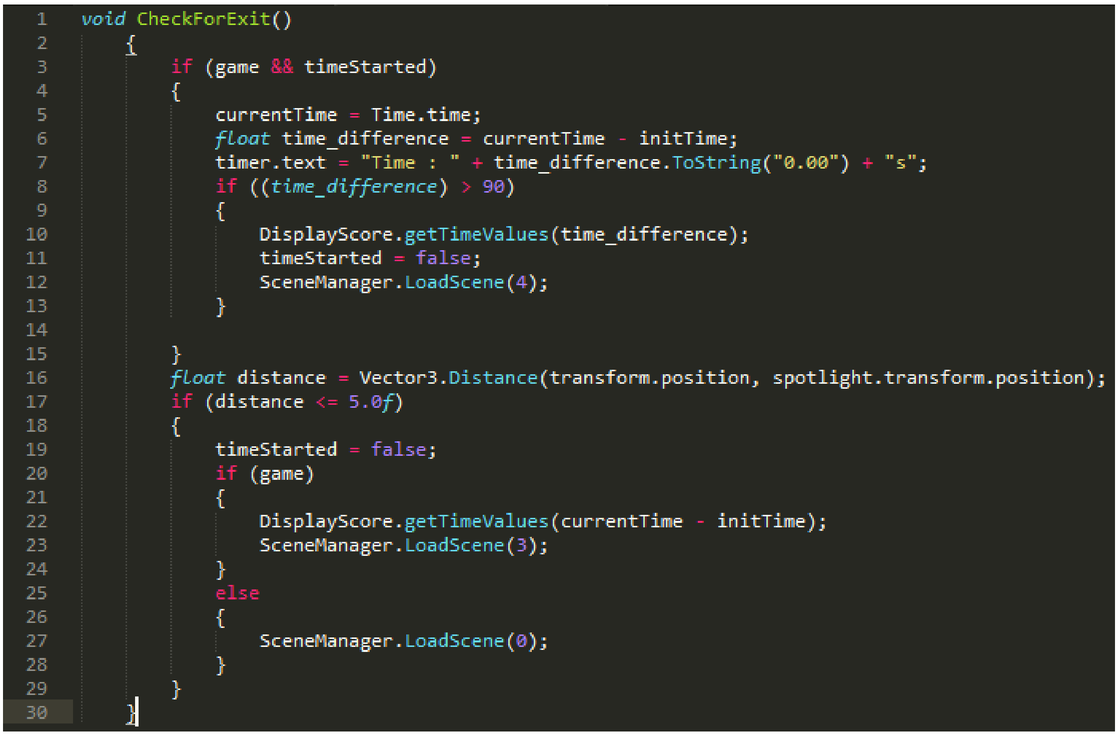
- (1-2)->3->(4-5)->6->11->3->(12-13)

  Testcase: room.Count = 1, distance > 10

Expected Output: return description of nearest room

Observed Output: returns empty string

1.8.2 Funtion: CheckForExit()



```
void CheckForExit()
    {
        if (game && timeStarted)
        {
            currentTime = Time.time;
            float time_difference = currentTime - initTime;
            timer.text = "Time : " + time_difference.ToString("0.00") + "s";
            if ((time_difference) > 90)
            {
                DisplayScore.getTimeValues(time_difference);
                timeStarted = false;
                SceneManager.LoadScene(4);
            }

        }
        float distance = Vector3.Distance(transform.position, spotlight.transform.position);
        if (distance <= 5.0f)
        {
            timeStarted = false;
            if (game)
            {
                DisplayScore.getTimeValues(currentTime - initTime);
                SceneManager.LoadScene(3);
            }
            else
            {
                SceneManager.LoadScene(0);
            }
        }
    }
```

Figure 1.33: Code for CheckForExit() function

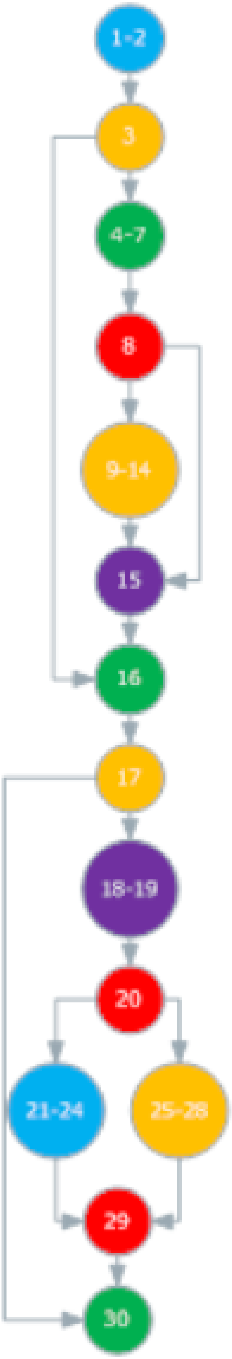


Figure 1.34: CFG for CheckForExit() function

1.8.2.1 Calculation of Linearly Independent Paths

Number of Linearly independent paths = Number of Edges - Number of nodes + 2 = 5

1.8.2.2 Linearly Independent Paths

- (1-2)->3->(4-7)->8->(9-14)->15->16->17->(18-19)->20->(23-24)->29->30

  Testcase: game = True, timeStarted = True, time_difference ¿ 90, distance ¡= 5

  Expected Output: Load the DisplayScore Scene and ends game

  Observed Output: Load the DisplayScore Scene and ends game

- (1-2)->3->(4-7)->8->(9-14)->15->16->17->30

  Testcase: game = True, timeStarted = True, time_difference ¿ 90, distance ¡= 5

  Expected Output: Load the DisplayScore Scene and ends game

  Observed Output: Load the DisplayScore Scene and ends game

- (1-2)->3->16->17->30

  Testcase: (game & timeStarted) =False, distance > 5

  Expected Output: Do nothing

  Observed Output: Do nothing

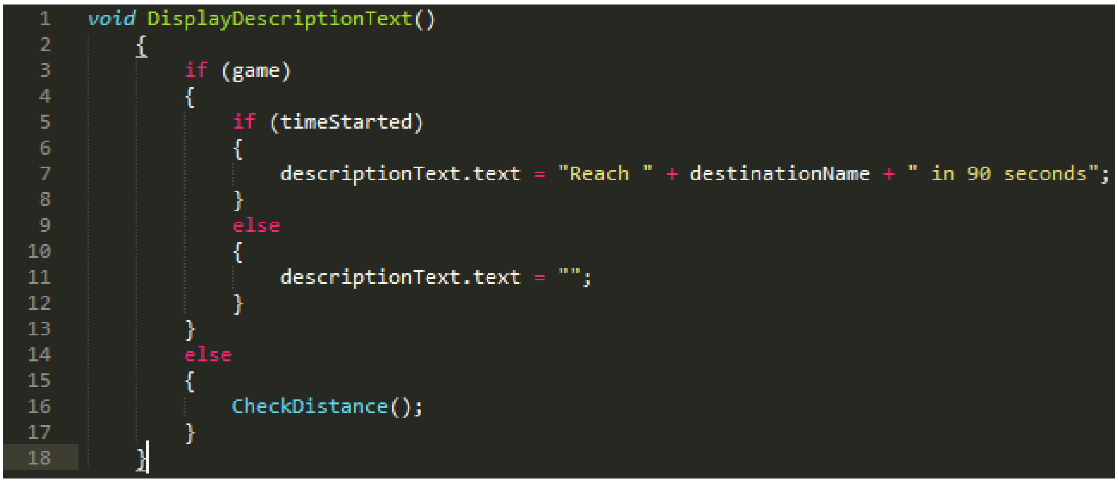
- (1-2)->3->(4-7)->8->15->16->17->30

  Testcase: game = True, timeStarted = True, time_difference ¡= 90, distance ¿ 5

  Expected Output: Update Timer text box

  Observed Output: Update Timer text box

1.8.3 Funtion: DisplayDescriptionText()



```
1   void DisplayDescriptionText()
2       {
3           if (game)
4           {
5               if (timeStarted)
6               {
7                   descriptionText.text = "Reach " + destinationName + " in 90 seconds";
8               }
9               else
10              {
11                  descriptionText.text = "";
12              }
13          }
14          else
15          {
16              CheckDistance();
17          }
18      }
```

Figure 1.35: Code for DisplayDescriptionText() function

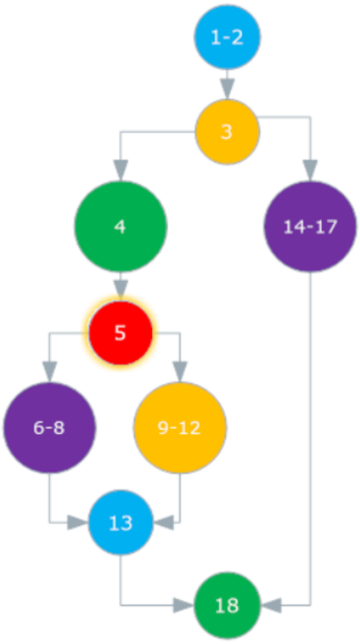


Figure 1.36: CFG for DisplayDescriptionText() function

1.8.3.1 Calculation of Linearly Independent Paths

Number of Linearly independent paths = Number of Edges - Number of nodes + 2 = 3

1.8.3.2 Linearly Independent Paths

- (1-2)->3->4->5->(6-8)->13->18

  Testcase: game = True and timeStarted = True

  Expected Output: Displays initial instructions for the game

  Observed Output: Displays initial instructions for the game

- (1-2)->3->4->5->(9-12)->13->18

  Testcase: game = True and timeStarted = False

  Expected Output: Makes the description textbox empty

  Observed Output: Makes the description textbox empty

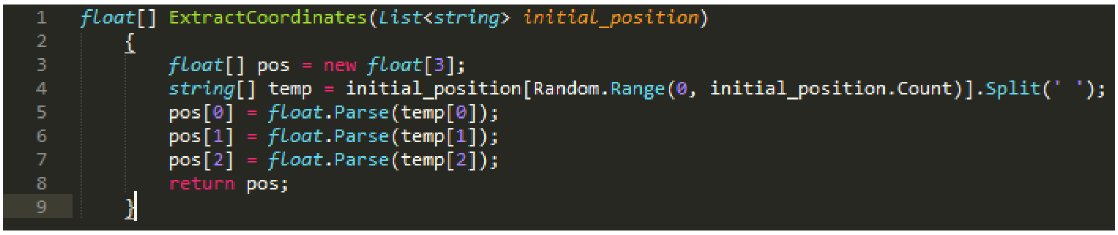
- (1-2)->3->(14-17)->18

  Testcase: game = False

  Expected Output: Display the name and Description of rooms in a radius of 10 units

  Observed Output: Display the name and Description of rooms in a radius of 10 units

1.8.4 Funtion: ExtractCoordinates()



```
float[] ExtractCoordinates(List<string> initial_position)
    {
        float[] pos = new float[3];
        string[] temp = initial_position[Random.Range(0, initial_position.Count)].Split(' ');
        pos[0] = float.Parse(temp[0]);
        pos[1] = float.Parse(temp[1]);
        pos[2] = float.Parse(temp[2]);
        return pos;
    }
```

Figure 1.37: Code for ExtractCoordinates() function

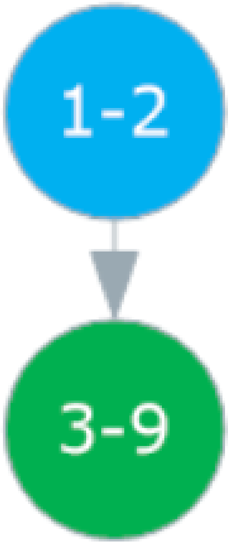


Figure 1.38: CFG for ExtractCoordinates() function

1.8.4.1 Calculation of Linearly Independent Paths

Number of Linearly independent paths = Number of Edges - Number of nodes + 2 = 1

1.8.4.2 Linearly Independent Paths

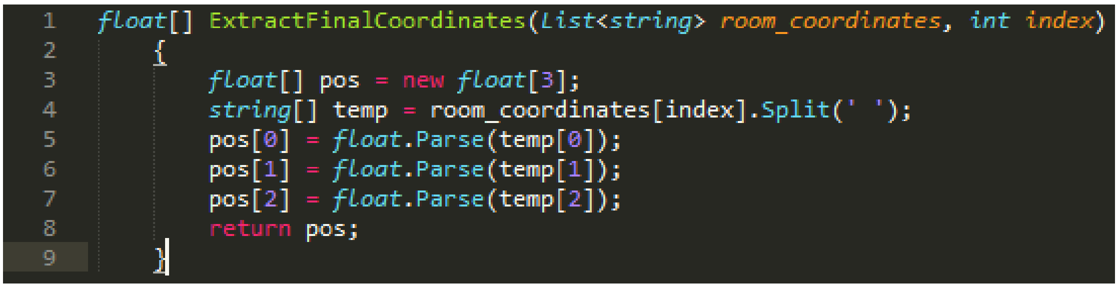
- (1-2)->(3-9)

  Testcase: all test cases

  Expected Output: extracts and return x,y,z coordinates (floats)

  Observed Output: extracts and return x,y,z coordinates (floats)

1.8.5 Funtion: ExtractFinalCoordinates()



```
float[] ExtractFinalCoordinates(List<string> room_coordinates, int index)
    {
        float[] pos = new float[3];
        string[] temp = room_coordinates[index].Split(' ');
        pos[0] = float.Parse(temp[0]);
        pos[1] = float.Parse(temp[1]);
        pos[2] = float.Parse(temp[2]);
        return pos;
    }
```

Figure 1.39: Code for ExtractFinalCoordinates() function

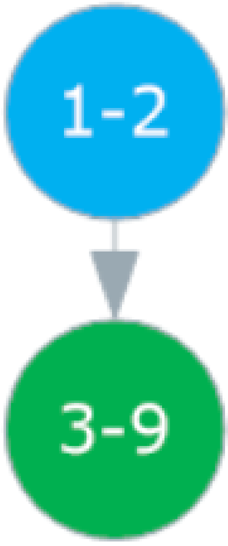


Figure 1.40: CFG for ExtractFinalCoordinates() function

1.8.5.1 Calculation of Linearly Independent Paths

Number of Linearly independent paths = Number of Edges - Number of nodes + 2 = 1

1.8.5.2 Linearly Independent Paths

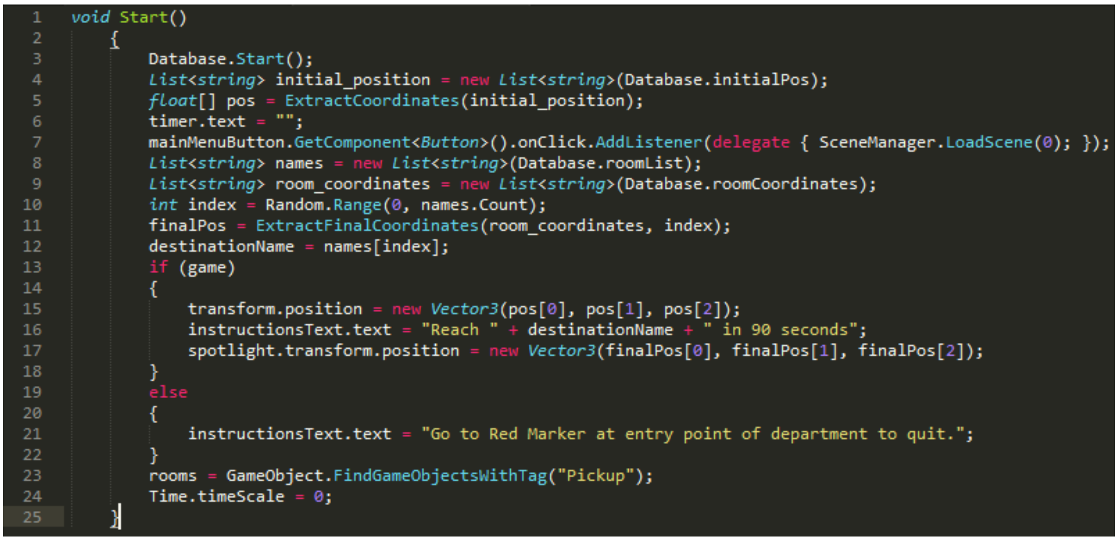
- (1-2)->(3-9)

  Testcase: all test cases

  Expected Output: extracts and return x,y,z coordinates (floats)

  Observed Output: extracts and return x,y,z coordinates (floats)

1.8.6 Funtion: Start()



```
 1  void Start()
 2      {
 3          Database.Start();
 4          List<string> initial_position = new List<string>(Database.initialPos);
 5          float[] pos = ExtractCoordinates(initial_position);
 6          timer.text = "";
 7          mainMenuButton.GetComponent<Button>().onClick.AddListener(delegate { SceneManager.LoadScene(0); });
 8          List<string> names = new List<string>(Database.roomList);
 9          List<string> room_coordinates = new List<string>(Database.roomCoordinates);
10          int index = Random.Range(0, names.Count);
11          finalPos = ExtractFinalCoordinates(room_coordinates, index);
12          destinationName = names[index];
13          if (game)
14          {
15              transform.position = new Vector3(pos[0], pos[1], pos[2]);
16              instructionsText.text = "Reach " + destinationName + " in 90 seconds";
17              spotlight.transform.position = new Vector3(finalPos[0], finalPos[1], finalPos[2]);
18          }
19          else
20          {
21              instructionsText.text = "Go to Red Marker at entry point of department to quit.";
22          }
23          rooms = GameObject.FindGameObjectsWithTag("Pickup");
24          Time.timeScale = 0;
25      }
```

Figure 1.41: Code for Start() function

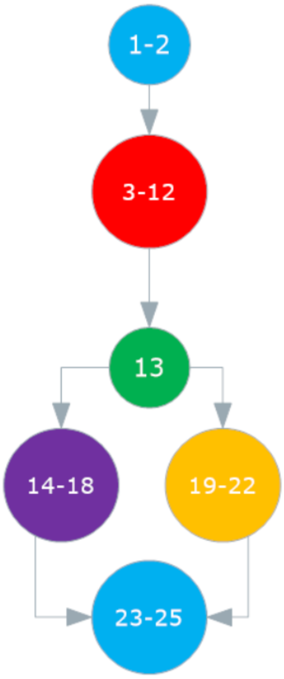


Figure 1.42: CFG for Start() function

1.8.6.1 Calculation of Linearly Independent Paths

Number of Linearly independent paths = Number of Edges - Number of nodes + 2 = 2

1.8.6.2 Linearly Independent Paths

- (1-2)->(3-12)->13->(14-18)->(23-25)

  Testcase: game = True

  Expected Output: Display Instructions message (for new game) on the screen.

  Observed Output: Display Instructions message (for new game) on the screen.

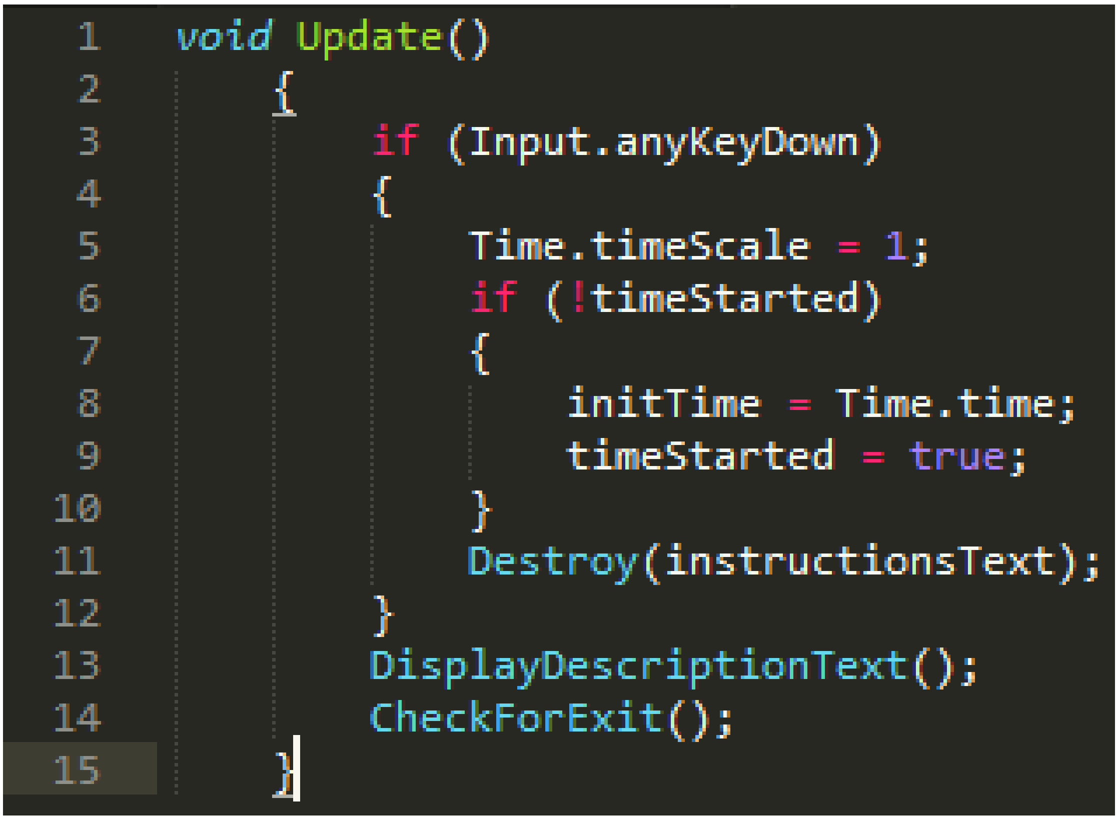
- (1-2)->(3-12)->13->(19-22)->(23-25)

  Testcase: game = False

  Expected Output: Display Instructions message (for training) on the screen.

  Observed Output: Display Instructions message (for training) on the screen.

1.8.7 Funtion: Update()



```
void Update()
    {
        if (Input.anyKeyDown)
        {
            Time.timeScale = 1;
            if (!timeStarted)
            {
                initTime = Time.time;
                timeStarted = true;
            }
            Destroy(instructionsText);
        }
        DisplayDescriptionText();
        CheckForExit();
    }
```

Figure 1.43: Code for Update() function

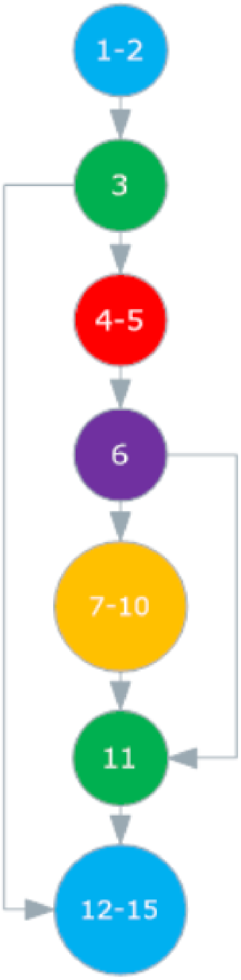


Figure 1.44: CFG for Update() function

1.8.7.1 Calculation of Linearly Independent Paths

Number of Linearly independent paths = Number of Edges - Number of nodes + 2 = 3

1.8.7.2 Linearly Independent Paths

- (1-2)->3->(4-5)->6->(7-10)->11->(12-15)

  Testcase: Input.anyKeyDown = True (a key is pressed) and timeStarted = False

  Expected Output: Destroys instruction text textbox and updates description textbox

Observed Output: Destroys instruction text textbox and updates description textbox

- (1-2)->3->(12-15)

  Testcase: Input.anyKeyDown = False (No key is pressed)

  Expected Output: updates description textbox and calls CheckForExit().

  Observed Output: updates description textbox and calls CheckForExit().

- (1-2)->3->(4-5)->6->11->(12-15)

  Testcase: Input.anyKeyDown = True (a key is pressed) and timeStarted = True

  Expected Output: updates description textbox and calls CheckForExit().

  Observed Output: updates description textbox and calls CheckForExit().