Middle East Technical University

Department of Computer Engineering

CENG 492

Computer Engineering Design II

Test Specification Plan

for TraffEdu

CodeSchbeke Software Solutions

Project Title: EDU3D

Advisor Asistant: Onur Soysal

Project Members: *Sıla Arslan*
Çağla Okutan
Hatice Kevser Sönmez
Bahar Pamuk
Ebru Doğan

SPRING 2006

TABLE OF CONTENTS

1) Introduction

The purpose of this document is to present the scope, strategies and schedule of testing which will be fulfilled in one-month time period after the first release of TraffEdu system of CodeSchbeke Software Solutions. In this test specification, we aimed to check and provide the consistency of the modules and the communication of the modules when they are integrated in the sense that we provide a bug-free and a well built project. The testing plan of this project is tried be developed adequate for an editor and a 3D animation.

2) Testing of the Modules

The TraffEdu system consists of 7 different modules; System Data Structures, GUIEventHandler, GUIEventHandler, OSGWindows, AnimationWindow, PhysicsEngine, FileHandler and Audio. We are aimed to perform both white box and black box testing for most of the modules but perform only black box testing to test integration of the modules.

In the white box testing, the members will focus on the program structure, coding details and integration of the modules. The white box testing will be left to the developer of the modules, who will consider the coding standards, the unused variables, the memory allocations for pointers which require internal knowledge of software.

Black box testing is performed by the members who are not the developer of the project. The developer will not focus on the internal structure of the modules, but on testing the functional requirements. Some sample black box tests are given in this document.

It should also be noted that, in case of a failure in any white box test, all the black box tests will be repeated. Thus white box tests will be performed by the developer of the module before any black box test is performed. One more thing is that, schedule at the end of the document shows the testing schedule of black box tests, not the white box tests which will be implemented and performed beforehand.

3) System Data Structures

In TraffEdu, there are many kinds of objects that user may add into the scene for creating the traffic case scenarios; these are mainly vehicles, roads, traffic lights, traffic signs and environmental objects with different models for each object type. Also, the properties of each object may differ from each other. However, there is a base class called *TraffEduObject* from which the other object types are derived from. Other than the fundamental data structures mentioned above, there is one more data structure, "Road", which does not derive from *TraffEduObject* class and has quite different properties than the other objects. So, the way that we test the "Road" will be different compared to the other fundamental data types.

For objects other than the road, essential points are, whether the positions and the rotations of the objects can be adjusted correctly and the models of the objects can be loaded. For "Road", the followings will be tested; the positions of the points that the user determines, the curve that is drawn by using those points, shape of the road, and texture of the road.

Sample Test Scenarios:

1) test scenario1
    - click on "Traffic Lights" button on the ToolBox and click on any traffic light type button
    - click on any point on the scene where you want to see the traffic light
    - see the correct traffic light is created at the position where the mouse touched
    - right click on the traffic light
    - click rotate on the right click menu
    - see that the traffic light is rotated

The same scenarios are also valid for the other object types (traffic sign, vehicle, environmental object) and will be repeated for each of them. For "Road" testing scenario will be different.

2) test scenario2
   - click on the “Roads” button on the ToolBox and click on any road type button
   - in the scene, click on the points the road will pass through
   - see the points (control points) that the mouse clicked are marked with a red sign
   - see the curve that is drawn by using those control points
   - click on the “Texturize” button on the ToolBox
   - see if the road type selected from the toolbox is created
   - see if the road passes through the control points
   - click on “Show/Hide Lines” button on the ToolBox
   - see curve and the points of the road are hidden
   - click on “Show/Hide Lines” button on the ToolBox again
   - see curve and the points of the road are displayed

4) GUIEventHandler Module

As mentioned in the previous reports, TraffEdu system has two main components and one of them is the TraffEdu Editor. Main work-flow of the TraffEdu Editor is getting inputs from the user and operating accordingly. As a result, there are many kinds of events that should be handled in the system.

In order to satisfy these needs, TraffEdu Editor will be tested by using both Black Box testing and White Box testing. Black Box testing will be used because it will discover faults of omission, indicating that part of the specification has not been fulfilled. White Box testing will also be used because it will discover faults of commission, indicating that part of the implementation is faulty. As stated earlier white-box tests are left to the developer.

Tester will control activities and changes occur concurrently in the menubar, toolbar, toolbox and OSG window when she applies test scenarios. Also tester will control whether those activities and changes are in the correct order as specified in the design phase.

Sample Test Scenarios:

1) test scenario1
    - click insert → object → vehicle → vehicleTypeX
    - see a vehicle in vehicleTypeX type is created in the default position
    - click insert → object → traffic sign → signTypeX
    - see a traffic sign in signTypeX type is created in the default position
    - click edit → selectAll
    - see the vehicle and the traffic sign are highlighted
    - click edit → delete
    - see the vehicle and the traffic sign are deleted from the scene
    - click edit → undo
    - see the vehicle and the traffic sign appear as highlighted
    - click edit → redo
    - see the vehicle and the traffic sign are deleted from the scene

2) test scenario2
    - click insert → object → traffic light
    - see a traffic light is created in the default position
    - drag&drop the traffic light to another position
    - click insert → object → sign → signTypeX
    - see a traffic sign in signTypeX type is created in the default position
    - click insert → object → traffic light
    - see another traffic light is created in the default position
    - click view → hide/display → traffic lights
    - see all the traffic lights in the scene are hidden

5) OSGWindows Module

OSGWindows module includes *OSGEventHandler* and *TimelineEventHandler* classes. Similar to the GUIEventHandler module, this module handles inputs but coming from OSG windows (main window and timeline window) not from menubar, toolbar or toolbox. Similar testing methods with the GUIEventHandler module will be used in this module

The tester will try possible mouse and key combinations that are recognized by the program and observe whether expected consequence is yielded.

Sample Test Scenarios:

1) test scenario1
    - click insert → object → vehicle → vehicleTypeX
    - left click on the vehicle
    - see the vehicle is highlighted
    - make drag&drop operation
    - see position of the vehicle is changed in the desired way
    - click insert → object → traffic light
    - see a traffic light is created in the default position
    - press CTRL key and left click on the traffic light
    - see the traffic light is highlighted too (multiple selection)

2) test scenario2
    - click insert → object → traffic light
    - see a traffic light is created in the default position
    - right click on the traffic light
    - see a right click menu appears on the right bottom corner of the object
    - click delete on the right click menu
    - see the traffic light is deleted from the scene

3) test scenario3
    - click on any quad on the timeline (frame1)
    - click frame → insert frame
    - click on the “vehicles” button on the ToolBox and click on any vehicle type
    - click on any point on the editor
    - see the correct vehicle type is created on the clicked position (frame1 position)
    - click on any other quad on the timeline (frame2)
    - click frame → insert frame
    - drag & drop the vehicle from the old position to the new position (frame2 position)
    - click on frame1 and see the vehicle is on the frame1 position
    - click on frame2 and see the vehicle is on the frame2 position

6) AnimationWindow Module

This class is directly related to the changes made in the editor, so it can not be checked independent from the editor. The frames that the user inserted in the timeline are called the keyframes and these are kept in a data structure in the program. The in-between frames are calculated and filled into another data structure and the program uses this structure for animation.

In animation, the essential point that needs to be checked is that, whether the positions and rotations of vehicles in the keyframes are the same with time intervals corresponding to these keyframes in the animation. Also, since the color of the traffic lights may be changed frame to frame, the color parameters should also be checked for the keyframes.

For vehicles, this will be done by comparing the position and rotation parameters in the data structure keeping the data of the key frames and the position of the vehicles for the keyframes in the animation. Tester will create a temporary file in the animation and write position parameters of vehicles for certain time units into this file. Then she will compare the position and rotation values in this file with the ones in the data structure keeping the keyframes.

For traffic lights, the color values (red, yellow, green) will be compared in the same way. The color parameters will be written to a temporary file during the animation for the time units corresponding to the keyframes. Thus, the values in the data structure keeping the keyframes and the values in this file are compared.

7) PhysicsEngine Module

For the animation part, the user has two options, one is determining the path of the vehicle by the keyframes in the process of creating a scenario and just watch the movement of the car and second is controlling the vehicle by the keyboard buttons without determining the path of the vehicle which is set previously in the editor.

For the first case, where the path of the vehicle is determined in the scenario, ODE is not used for the adjustments of speed and steer parameters of the vehicle. ODE is used only for collision detections. So, if a collision case is prepared in the editor, what is to be checked is whether this collision is happened in the animation.

For the second case, where the user controls the vehicle, again we check for the speed and steer but not by writing to a file but by checking that if the vehicle fulfills the action that the user enters as input from the keyboard or not.

8) FileHandler Module

The scenarios created by the user in the editor of TraffEdu are saved to a file by the FileHandler module. The saved file includes the properties, positions and rotations of each object for each frame and can be used for running the animation particular to that scenario. The file operations are done via the Menubar options under File Menu and Toolbar buttons.

When TraffEdu program is run, a new and default environment in the editor is created. After that user may add objects to the new environment and create a new scenario, or open an already saved file and run its animation. Also user can make some changes on the already saved scenario, then save and watch the new animation.

Sample Test Scenarios:

1) test scenario1
   - click on “New” button on the Toolbar
   - see a new environment is opened in the TraffEdu Editor
   - create any scenario in the editor
   - click on “Exit” button in the Toolbar
   - see the exit menu is asking for “Save” or “Exit”
   - click on “Save As” button
   - see the “Save As” menu asking for the filename
   - enter a filename and save
   - click on the animate button on the toolbar
   - see the animation is in the way that the scenario created in the editor

2) test scenario2
   - click on “New” button on the Toolbar
   - see a new environment is opened in the TraffEdu Editor
   - create any scenario in the editor
   - click on “New” button on the Toolbar again
   - see the warning window asking “Do you want to save changes?”
   - click on the “Yes” button

   - see a new environment is opened in the editor
   - see the old scenario is saved in the file whose name is entered with the opened save menu

9) Audio Module

As it is mentioned in the design report, TraffEdu system has some audios like speaking of the user inserted in the TraffEdu Editor and sound effects in the animation. Audio module controls these kinds of sounds. A number of black box tests will be performed on the Audio module to test whether desired sounds are played on the desired time during the desired duration.

Sample Test Scenarios:

1) test scenario1
   - prepare a scenario which includes more than one vehicle
   - in one of the frames make two vehicles located in the same position (that means a collision will occur)
   - click animate button on the menubar
   - sound effect should be heard when two cars are crashed

2) test scenario2
   - prepare frames for any animation
   - in one frame, click insert → audio → select a sound file name (frameX)
   - click animate button on the menubar
   - sound / speech should be started to play at the time interval of the frameX

10) Integration Tests

In order to control all the modules work properly in the system when they are integrated under the same project, the team will have to make scenario based tests that control the paths specified in the use case diagrams, within the design. We will cover these tests under the topic "Integration Tests". These will be high level, Black-Box tests with scenario specific inputs and expected results given and look for the specified outputs for the test case.

Sample Test Scenarios:

1) test scenario1 (Integration test for GUIEventHandler, OSGWindows, System Data Structures, Audio, AnimationWindow and PhysicsEngine)
   - click on any quad on the timeline (frame1)
   - click frame → insert frame
   - click on the "vehicles" button on the ToolBox and click on any vehicle type button
   - click on any point on the editor
   - see the correct vehicle type is created on the clicked position (vehicle1)
   - click on the "vehicles" button on the ToolBox and click on any vehicle type

button

- click on any other point on the editor
- see the correct vehicle type is created on the clicked position (vehicle2)
- click on any other quad on the timeline (frame2)
- click frame → insert frame
- change position of both vehicles from their old positions to a new position
- click animate button on the menubar
- see both vehicles are at the same position on the frame2
- see vehicles do not penetrate from each other
- sound effect should be heard when the vehicles are crashed

2) test scenario2 (Integration test for GUIEventHandler, OSGWindows, System Data Structures)

- click on any quad on the timeline (frame1)
- click frame → insert frame
- click insert → object → traffic light
- see a traffic light, which has green color, is created in the default position
- click on any other quad on the timeline (frame2)
- click frame → insert frame
- double click on the traffic light
- see color of the traffic light is changed from green to yellow
- click on frame1
- see color of the traffic light is green
- right click on the traffic light
- click delete on the appeared right click menu
- see the traffic light is deleted from frame1
- click on frame2
- see the traffic light is deleted from frame2 too

11) Testing Schedule

Module Name	Tester(s)	Start Date	End Date	Duration (days)
System Data Structures	Ebru	15.05.2006	18.05.2006	3 days
GUIEventHandler	Kevser	15.05.2006	20.05.2006	5 days
OSGWindows	Bahar	15.05.2006	20.05.2006	5 days
AnimationWindow	Ebru, Bahar	18.05.2006	23.05.2006	5 days
PhysicsEngine	Kevser	20.05.2006	24.05.2006	4 days
FileHandler	Çağla	15.05.2006	19.05.2006	4 days
Audio	Sıla	15.05.2006	18.05.2006	3 days
Integration Tests	All	18.05.2006	27.05.2006	9 days

12) Conclusion

In this test specification report we presented the test methods we will use, testers assigned to that tests and the schedule for the developers for each module of the project. The number of test scenarios depends on the module and in each module our goal is to recover as much errors as we can. Some white box tests are done concurrently with the implementation of the modules so in this document we especially concentrated on the black box tests.