# DiRT – Dust in Real-time: The Specification Process

**Marcos Bagby, Ryan Romero, Brett Sulprizio, Hiroko Uda**
**Joseph Jaquish, and Frederick C. Harris, Jr.**

Department of Computer Science and Engineering
University of Nevada, Reno
Reno, NV, 89557, USA
{mbagby, rromero, sbrett, uda, jaquish, fredh}@cse.unr.edu

**Abstract** *Dust in Real-time (DiRT) is a 3D dust visualization program and interactive computer benchmarking tool designed to model real world dust dynamics in a virtual environment, in real-time. The interactive benchmarking tool allows users to track and gauge the performance of their system's ability to render the dust by issuing real-time system and environment related reports. The system reports update, in accordance with the user's quality and realism settings. Similar to a video game, users will be able to "play" during the simulation by way of a simple vehicle simulator. This paper presents details of DiRT's UML requirements specification, software architecture specifics, high and low-level design details, user interface principles and snapshots.*

**Keywords:** Dust, real-time computation, virtual environment, benchmarking, requirements specification, software architecture.

## 1. Introduction

Dust can be defined as small, dry particles of matter that are hardly noticeable unless present in large quantities. Dust-storms and dust-devils are small examples of atmospheric agitation of the terrain in which the soil is disturbed and drawn up into a dust cloud. Dust during this behavior can be quite harmful, even incapacitating to people or equipment. Understanding the behavior and effects of dust has significant military applications, thus the modeling and visualization of dust can be beneficial. The modeling of dust for use in simulations may help modern combat experts plan for or pre-empt the affects of dust caused by vehicles or the atmosphere. A better understanding of the combat environment will lead to better planning and more effective strategy.

Typically the visualization of dust (or some environmental phenomenon) involves the use of large particle engines and complex physics equations which control those particles. Most computers cannot render dust intensive environments using this method in real-time. An environment rendered in real-time implies that a user can navigate through the environment and each image is rendered very quickly. This gives the user the flexibility to travel wherever need be and it helps make the simulation as realistic as possible. But to get an accurate picture of a dust cloud (i.e.: accurate according to real world physics), the computation is very time consuming. A user would not be able to navigate through an environment without long pauses between the rendering of each image. The common method used to visualize realistic physics is to render each image and store it, then once all of the images have been generated, create a video from these. Though realistic, the user has no ability to navigate through the environment. Thus, our objective is to find a method which will realistically render dust behavior without any resource intensive particle engine.

DiRT is a 3D dust visualization program which includes interactive benchmarking capabilities. The system is designed to model dust as realistically as possible using as few system resources as possible. The program will utilize real-time lighting and shadows to show the affects of light through a dust cloud as well as dust modeling by way of diffuse reflection, volumetric fog, and an optimized particle engine. Changes in dust activity and density will be directly related to the soil type and environmental conditions. Multiple camera views will be available to witness the dust from a user chosen position.

The authors intend to port the dust visualization algorithm to a CAVE (Cave Assisted Virtual Environment). A CAVE is a 3D virtual environment. When a person looks at the CAVE projection screens with 3D glasses on, everything appears to be three dimensional. When surrounded by enough screens, one gets the perception of being immersed in that environment. Thus a CAVE is referred to as an immersive virtual environment. Running our algorithm in a CAVE will allow users to experience the behavior of dust more realistically than a single two dimensional screen.

Natural dust storms and dust activity caused by man-made machines are extremely intricate inertial systems in which the most minute detail can be responsible for the greatest change in behavior. Friction and gravity are two significant forces at work, but other aspects such as particle size, weight, material composition, not to mention barometric pressure, humidity and precipitation are all factors that affect dust and its behaviors. These aspects must be taken into consideration but with varying degrees of priority per the computer's resource usage.

The remainder of this paper is structured as follows: Section 2 describes the functional and nonfunctional software requirements for the DiRT system. Section 3 presents the use case diagram of DiRT. Section 4 describes the software's high-level architecture. Section 5 provides details of its medium and low-level design. Section 6 reports on DiRT's current status and points to future work. Finally, Section 7 contains the authors' closing remarks.

## 2. Requirements Specification

Using the notations presented in [2], the requirements for DiRT, structured as functional and non-functional requirements, are specified in the next two subsections.

### 2.1 Functional Requirements

Table I contains functional requirements with priority levels 1-3 indicated in square brackets (1 being the most important).

### 2.2 Non-Functional Requirements

Table II contains a list of the non-functional requirements that the dust modeling program shall fulfill or hope to fulfill.

## 3. Use Case Modeling

To better understand DiRT's functionality, the system has been divided into use cases. Section 3.1 presents the use case diagram, shown in Figure 1, which shows interaction between the user and DiRT. Section 3.2 gives the detailed use cases, and Section 3.2 details the Requirements Traceability Matrix shown in Figure 2.

**R01** [1] DiRT shall use volumetric fog extensions.
**R02** [1] DiRT shall simulate dust dissipation.

**R03** [1] DiRT simulates wind patterns.

**R04** [3] DiRT shall check to see if current computer hardware can run the simulation.
**R05** [2] DiRT shall display the current frame rate.
**R06** [1] DiRT shall allow the user to start the simulation.
**R07** [1] DiRT shall allow the user to stop the simulation at any time.
**R08** [1] DiRT shall display the methods of dust simulation.
**R09** [3] DiRT shall allow the user to change the terrain.
**R10** [1] DiRT shall allow the user to change the point of view.
**R11** [3] DiRT shall enable the user to customize the lighting.
**R12** [1] DiRT shall support a free form camera.

**R13** [1] DiRT shall support a first person pilot camera view.
**R14** [3] DiRT shall allow the user to load terrain maps.
**R15** [2] DiRT enables the user to change the screen resolution.
**R16** [2] DiRT shall enable the user to switch from pilot view to free camera perspective.
**R17** [1] DiRT shall use the keyboard to control vehicle movement.
**R18** [1] DiRT shall enable the pilot to fly/drive the vehicle.
**R19** [2] DiRT shall enable the user to change the model used for the vehicle.
**R20** [3] DiRT shall enable the user to toggle the terrain to on, off, wire-frame, smooth, and smooth + wire-frame.
**R21** [3] DiRT shall enable the user to change the speed of the simulation (slow down or speed up).

**Table I.** DiRT Functional Requirements

**T01** DiRT will be able to run all dust methods in real-time.
**T02** DiRT will ask a user to choose a displaying method from the three:
1. Volumetric fog dust
2. Diffuse reflection dust.
3. Particle system.
**T03** DiRT shall be implemented with OpenGL and Qt.

**T04** DiRT shall be a cross-platform application.
**T05** DiRT shall be written in C++.

**Table II.** DiRT Non-Functional Requirements

### 3.1 Use Case Diagram

The use case diagram displayed in Figure 1 shows the interaction between a user and the DiRT program. The program will be useable from the instant all the textures and maps finish loading into memory. The user will be able to drive their vehicle in the free drive/flight mode. During this mode the player may customize the dust methods, video resolution, simulation speed, model type, and terrain.

### 3.2 Detailed Use Cases

To help further understand DiRT's functionality, detailed use cases are presented below.

**UC01**: *EnableFogDust* (Enable Volumetric Fog Dust) – Enables the user to simulate dust using the volumetric fog method.

**UC02**: *EnableRefractionDust* –Enables the user to simulate the dust using the light refraction method.

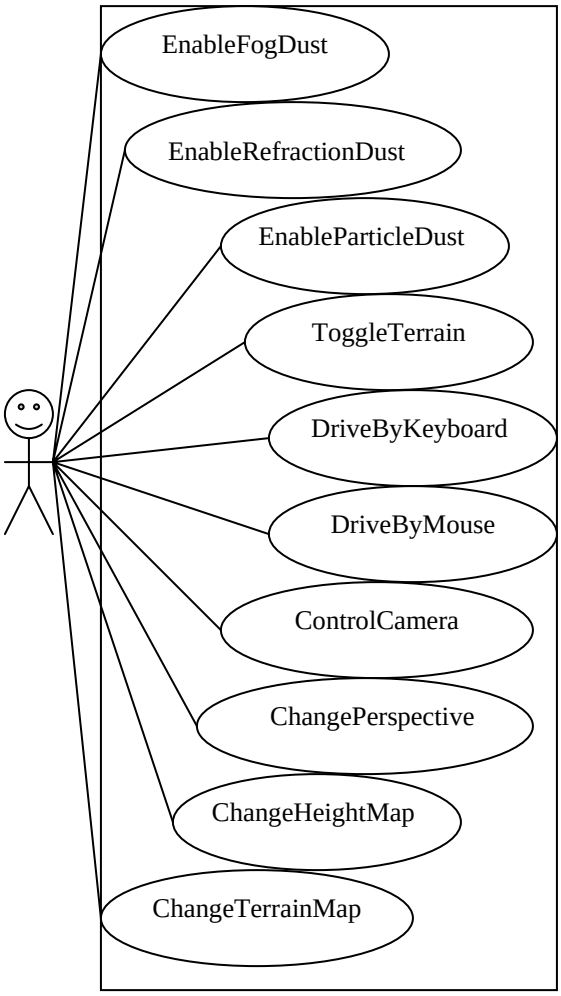


**Figure 1.** Use Case Diagram for DiRT

**UC03**: *EnableParticleDust* –Enables the user to simulate the dust using the particle dust method.

**UC04**: *ChangeDraft* –Allows the user to increase or decrease the amount of force of air currents.

**UC05**: *DrivebyMouse* –Allows the user to control the movement of the model by use of the mouse.

**UC06**: *DrivebyKeyboard* –Allows the user to control the movement of the model by use of the keyboard.

**UC07**: *ControlCamera* –Allows the user to see the simulation from different points of view by enabling camera movement.

**UC08**: *ChangeCameraPerspective* –Allows the user to change the camera perspective from pilot view to free camera view, and back again.

**UC09**: *ToggleTerrain* –Allows the user to choose between having and not having a terrain to run the simulation in.

### 3.3 Requirements Traceability Matrix

A partial requirements traceability matrix is shown in Figure 2 which depicts the mapping between the use cases and the functional requirements of the DiRT system. The complete requirements traceability matrix can be found in [11].

## 4. Architectural Design

The system architecture is divided into different subsystems as shown in Figure 3. Subsystem descriptions are below.

| | Use Case | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **01** | **02** | **03** | **04** | **05** | **06** | **07** | **08** |
| **R06** | X | | | | | | | |
| **R07** | | X | | | | | | |
| **R08** | | | | | | | | X |
| **R09** | | | | | | | | X |
| **R10** | | | | | X | | | |
| **R11** | | | | | | | | |
| **R12** | | | | | X | | | |
| **R13** | | | | | X | | | |
| **R14** | | | | | | X | | |
| **R15** | | | | | | | X | |
| **R16** | X | | | | | | | |
| **R17** | | X | | | | | | |

**Figure 2.** Requirements Traceability Matrix for DiRT (partial)

**GUI** The Graphical User Interface (GUI) subsystem contains all the Qt classes including Qt's OpenGL classes. The classes include GUI buttons, mouse and keyboard control schemes. It is also responsible for displaying system runtime information such as frame rate and memory usage.

**Simulation** The *Simulation* subsystem contains the main "game" engine that controls the free flight/drive aspect of the simulation, along with the simulation lighting and dust physics, and has complete control of the scene cameras. It is also the back end through which the GUI interfaces with the other subsystems.

**Terrain** The *Terrain* subsystem contains classes that are responsible for loading and drawing the particularly large terrain height maps as well as texturing.

**Model** The *Model* subsystem contains classes for loading and drawing the different vehicle models during simulation. It also contains the classes which control the model movement in response to user input commands.

**Dust Methods** The *Dust Methods* subsystem controls the 3 dust rendering engines. The fog engine morphs the fog volumes. The reflection engine sets the correct lighting over a similar volume. The particle engine controls a particle system which acts over the areas affected by the model.

## 5. Detailed Design

The structure of DiRT was done using an object oriented approach. As such, charts categorizing the particular program units as well as activity are given in subsections 5.1 and 5.2 respectively.

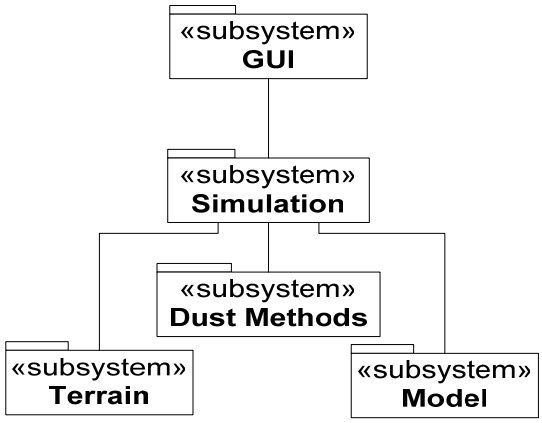


**Figure 3.** DiRT System Architecture

### 5.1 Class Diagram

A class diagram of DiRT showing the modularization of the program into object classes is presented in Figure 5. The diagram also includes details of operations, attributes, relationships, multiplicity constraints, and visibility for each class. Complete class and method descriptions can be found in [7]

### 5.2 System Activity Chart

In order to completely cover the design of DiRT, various diagrams were created as part of its software model, including activity charts. A sample activity chart of DiRT is presented in Figure 4.

## 6. Current Status and Future Work

The specification and design phases of the DiRT project have been completed. While some of the architectural components described in these documents may need slight modifications as the software is refined and completed, we have started implementation with a thoroughly and

critically reviewed concept. From the implementation work done so far, it became evident to us that the framework created is solid for driving the work ahead.

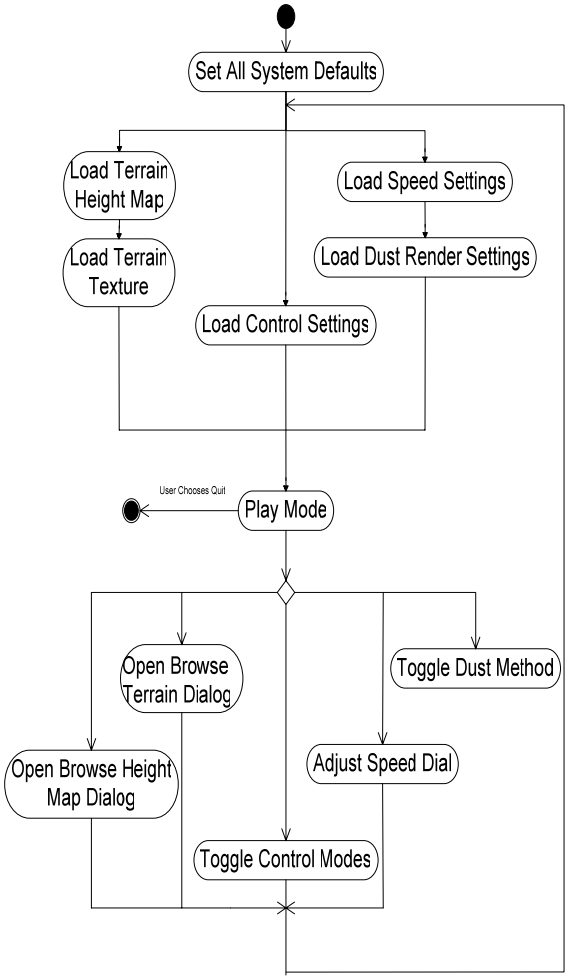


**Figure 4.** Activity Chart DiRT Displays the system operations during runtime.

## 7. Conclusions

The DiRT simulation whose specification has been presented in this paper is an innovative method of allowing generally resource intensive computing techniques to be substituted by less intensive methods yet still delivering similar results.

The focus of DiRT is the creation of a way for dust or wind-blow dirt or sand to be modeled and displayed visually in real-time.

## References


[1] Angel Edward., (2003) Interactive Computer Graphics: A Top-Down Approach Using OpenGL. *Pearson Education.*

[2] Blinn, James F., (July 1982) Light Reflection Functions for the Simulation of Clouds and Dusty Surfaces. *Jet Propulsion Laboratory, California Institute of Technology.*

[3] Chen, Jim X., et. al. (1999). Real-Time Simulation of Dust Behavior Generated by a Fast Traveling Vehicle. *ACM Transactions on Modeling and Computer Simulation, Vol. 9, No. 2,* 81-104. (Reference Paper)

[4] Lewis, Robert R. (February, 1998). Light-Driven Global Illumination with a Wavelet Representation of Light Transport. *University of British Colombia.*

[5] Martin, Allen., *Particle Systems* http://www.cs.wpi.edu/~matt/courses/cs563/talks/psys.html

[6] Fernando, Randima. (2004) *GPU Gems. Pearson Education.*

[7] DiRT Project, accessed March 19, 2005 at http://www.cs.unr.edu/~mbagby

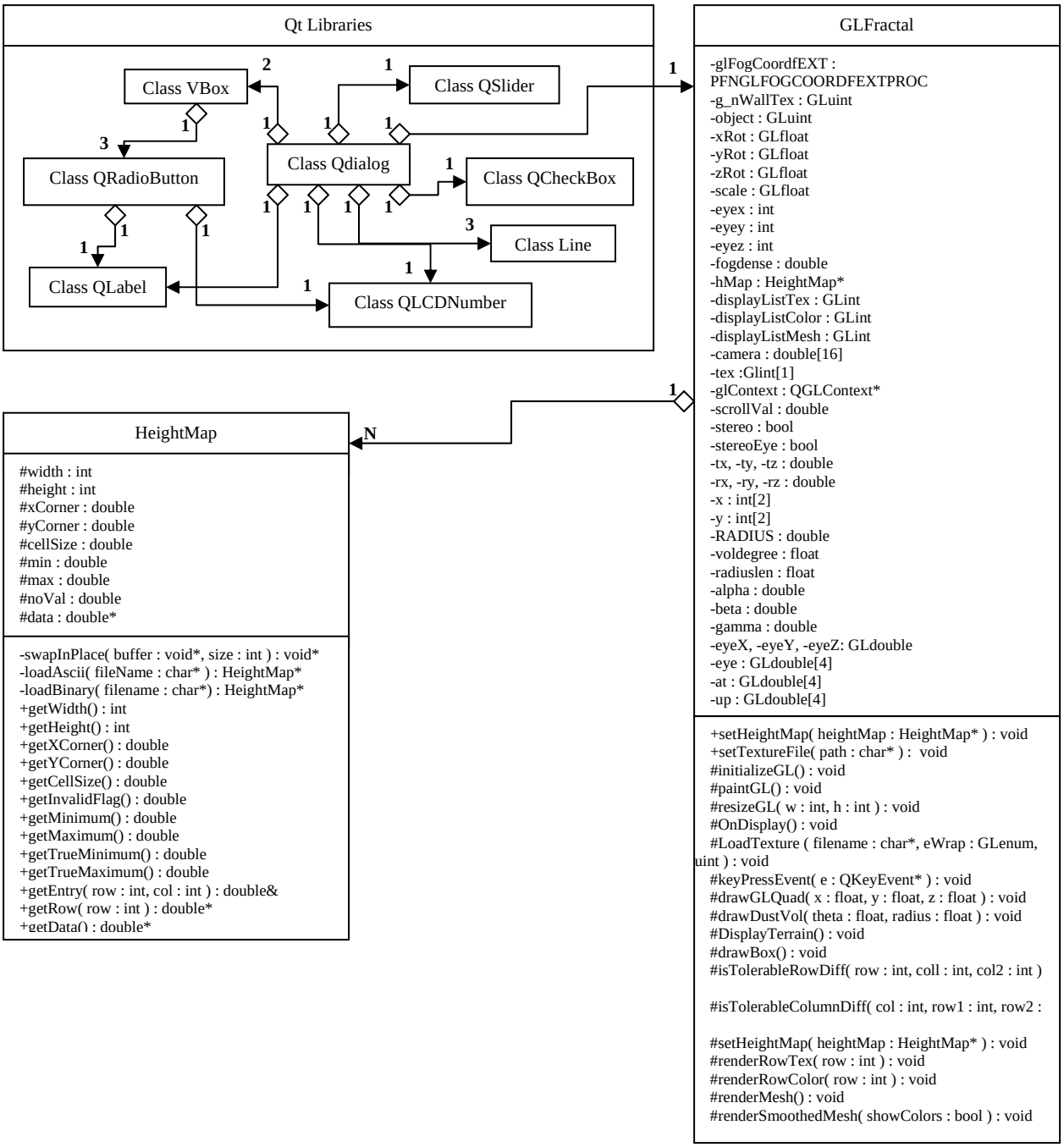


**Figure 5.** DiRT Class Diagram (partial)