SUED: An Extensible Framework for the Development of Low-cost DVE Systems

S. Casas*, P. Morillo[1], J. Gimeno[1] and M. Fernández[1]
Instituto de Robótica
Universidad de Valencia (Spain)
E-mail: Sergio.Casas@uv.es

ABSTRACT

The current expansion of 3D real-time applications based on collaborative virtual scenarios has facilitated the growth of distributed virtual environment (DVE) systems, where a set of remote users share a 3D virtual scene. Nowadays, the 3D graphics market does not ignore the presence of a new generation of demanding users who are driving the industry into a race to develop more and more complex and realistic simulations. This competition for the launch of new and innovative products, not only has caused the development costs to skyrocket, but also has motivated the development of incredibly complex platforms. In this paper, we present SUED, a new open-source software framework that has been intentionally designed to enable a rapid prototyping of low-cost DVE systems. Unlike other proposals, SUED offers two important features. On the one hand, it avoids complexity in programming and embeds some common tasks in the development of this type of systems by means of an easy-to-use interface based on XML files. On the other hand, it allows developers to extend the basic capabilities of the framework by defining custom modules on open-source and well-known Virtual Reality software tools. In addition, this paper presents two different DVE systems developed on SUED and the cost and time reductions achieved in their development.

Index Terms: I.3.7 [Three-Dimensional Graphics and Realism]: Virtual reality—; H.5.1 [Multimedia Information Systems]: Artificial, Augmented, and Virtual Realities—

1 INTRODUCTION

Distributed Virtual Environment (DVE) systems have experienced a spectacular growth last years. These realtime time systems allow multiple users, connected from different computers, to interact in a common 3D virtual world [9]. Although DVE systems have been traditionally used for multiplayer video games, military and industrial training, or collaborative engineering, these type of systems are continuously challenged by requirements for increasing complexity. This complexity used to be expressed in terms of system throughput (maximum number of users simultaneously connected to the system) and level of realism.

As the user requirements for a DVE system become more complex and distributed, the development costs raise a very serious concern about how to develop this type of systems efficiently [1]. Because of this reason, several software frameworks have been presented to minimize both time and complexity in the development of DVE systems [5, 7, 3, 10, 11, 4]. However, although the aim of these frameworks is to reduce the development costs and let DVE systems be implemented much faster, this cost reduction is not applied in a global manner. On the one hand, some approaches remain very linked to standard Virtual/Augmented Reality libraries [10, 11] or immersive visualization devices [7, 3]. On the other hand, more recent proposals either lack an easy configuration mechanism [5], or do not provide an unified mechanism to control the configuration and execution of the actions and events occurring during the simulations [4].

In this paper, we propose a new software platform called SUED that has been intentionally designed to enable a rapid prototyping of low-cost DVE systems. SUED models an entire simulation using a finite-state machine model and provides an easy-to-use interface for non technical users based on XML files. This approach reduces not only the development time of the DVE system, but also the complexity of the development when these systems are used to simulate complex scenarios or situations such as validation of emergency plans or security procedures. The modular architecture of SUED takes advantage of open-source and well-known Virtual Reality tools (in terms of low cost, reliability, and portability across platforms) and allows developers to easily extend the features of the framework.

The rest of the paper is organized as follows: Section 2 presents some related work focusing on the development of frameworks oriented to DVE systems. Then, Section 3 gives an overview and some design considerations of the SUED framework. Next, Section 4 describes two different DVE systems that have been developed using SUED. Finally, Section 5 outlines some concluding remarks and Section 6 points out the future work to be done.

2 RELATED WORK

Several software frameworks for the development of Virtual Reality (VR) or Augmented Reality (AR) applications have been proposed. A good review of this type of existing frameworks, including the most important references, is given in [2] and [6]. Although these references have had an important impact on the community of VR and AR developers, the developers of DVE systems have demanded custom solutions to address their particular functional requirements and research challenges.

Avocado [11] is an object-oriented system based on the Performer library where a number of shared-data structures are kept consistent among remote users by some global operations such as deletion or insertion. The system is open but remains very basic and still requires a substantial effort to build a new application [7]. JAPE [10] is a prototyping environment, based also on the Performer library, which has been built to support application design and algorithm development. Despite the fact it includes advanced features, such as tracking control or point-based objects, it still remains very linked to the blue-c project.

CAVEStudy [7] has been developed to take advantage of the rich interface that CAVE provides. However, this system is not totally collaborative since, when a group of remote users wish to collaborate during a data visualization session, they cannot interact with the 3D model simultaneously. A control scheme based on token

*Supported by the Spanish MEC under grant DPI2006-14928-C02-02 and the European Commission Network of Excellence INTUITION IST-NMP-1-507248-2

passing enables a simultaneous visualization but disables the modification of the model except for one specific user. A more advanced collaborative framework for CAVE environments is AR-CAVE [3]. AR-CAVE is a distributed multi-modal interaction platform that allows tangible and natural user interactions such as gesture-based inputs and tangible manipulations. However, it is specifically oriented to immersive visualization systems and cannot be considered as a general software framework for DVE systems.

Delta3D is a fully-functional open-source game and simulation engine appropriate for a wide variety of modelling and simulation applications [5]. Delta3D is close to game development and is a very thin API layer, which connects multiple OS projects such as OSG, ODE, CAL3D or OpenAL into a custom graphics engine. This integration of open-source and well-known VR software tools leads software developers to extend the framework features with low effort. However, Delta3D lacks both a single point of configuration and a variety of advanced network protocols. These deficiencies make the application development a longer process.

Finally, a recent component-based framework is MORGAN [4]. MORGAN is oriented to a rapid evaluation of VR and AR applications providing a distributed render engine with automatic scene graph synchronization capabilities. However, it is a commercial product that relies on a CORBA middleware for the network communication, which cannot be extended. As the rest of frameworks, MORGAN is not focused on the management and execution of complex simulations. Moreover, it does not provide a unified mechanism to define and control the simulation processes and actions performed in a complex 3D collaborative environment. Nowadays, this feature is a necessary condition to implement DVE systems much faster.

3 AN OVERVIEW OF SUED

SUED (which stands for **S**ystem for the development of **U**nexpensive and **E**xtensible **D**VE systems) is an open-source software platform designed to enable a rapid prototyping of low-cost DVE systems. Our framework is oriented to develop complex systems based on procedural simulations, which are represented following a finite-state machine model by means of some XML files. The software architecture of SUED is based on a modular approach oriented to develop procedural collaborative simulations. This approach allows expressing the simulation in terms of actions, states and transitions providing a powerful event-driven model.

One of the key features of SUED is that every single component is designed and (in special) implemented to be extended in an easy way. As a premise, if a software framework oriented to develop DVE systems wants to be useful, it needs to be both flexible (to serve for different purposes) and extendible (to be able to adapt to new requirements).

It is important to point out that although SUED is specifically designed to be the foundation of a procedural simulation, it is not limited to that kind of simulations, as it may perfectly suit any other type. In that case, some of the features included in SUED will not be taken advantage of, but it will still serve as a valid framework as it embeds many features any simulation needs to provide.

Needless to say, SUED will also work to build stand-alone graphic simulations rather than DVE systems. In this case, all the network architecture will be of no use whatsoever, but SUED will still be useful for it will allow the developers to save some precious time that will otherwise be *wasted* setting up an XML parsing interface, a procedural simulation module, stereo capabilities and a number of things that are repeatedly required when building a new simulation.

Furthermore, as SUED is able to detect this situation, it would disable all the network modules. This way, the system performance will not be compromised.

3.1 Software Architecture

SUED was designed in the form of what we like to call an "event-driven object-oriented architecture". That means all the components (modules) are designed following a classic object-oriented approach, but with the system working as a syncronous event-driven architecture. This is accomplished because all the objects are designed and built to evolute around the concept of 'event'. The system is not asynchronous because even though events can occur asynchronously, we capture them at discrete times.

Figure 1 shows how the architecture of SUED is organized as a set of components that are arranged following a modular approach. Each client of the DVE system runs a distinct and complete copy of the simulation identified by the "*KernelDVE*" module, which also provides a message-passing interface to communicate with the entire DVE system. This module includes a "*Networking*" submodule where the communication architecture [9] of the final developed DVE system can be configured. Although SUED incorporates by default an information exchange model based on a master/slave configuration, a networked-server or a P2P network protocol can be selected and properly configured. As the rest of modules contained within the architecture, more network protocols can be easily developed and added to SUED.

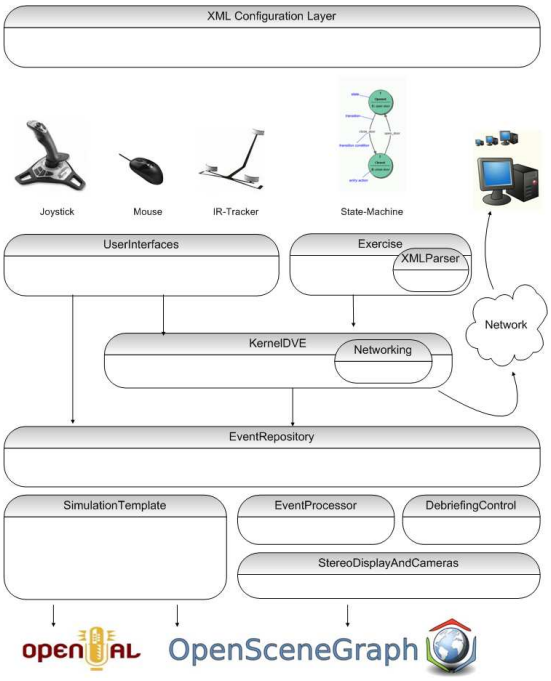

Figure 1: A modular view of the software architecture in SUED

SUED identifies each participant (client) of the simulation with an unique ID number ranging from 1 to N, being N the number of clients. The number of participants need not to be known at launch time, but unfortunately it can't (so far) change dynamically.

Events are then marked with a 'participant id' to identify which client they will be affecting to (i.e. which clients will be needing to execute them). Local events (events intended to be executed in the same machine they were generated) are identified with 'participant id = -1', meanwhile 'broadcast' events (events intended to be executed by all DVE clients) are identified with 'participant id = 0'. Multicast events are still not supported.

We think this convention is simple yet flexible enough to describe both DVE and stand-alone simulations, as well as different approaches for the DVE network protocols.

3.2 Implementation Decisions

The object-oriented design of SUED has been implemented using C++ as a back-end with open-source and well-known software libraries such as OSG, ODE, CAL3D, OpenAL and Xerces [8] as a front-end. The use of this type of C++ software libraries has allowed the development of SUED as a multiplatform framework to be executed on different operating systems (Win32/64, Linux and Mac OS X).

At first, some other implementation options were discussed, such as C# or managed C++, but they were discarded for a number of reasons. First of all, they lack the interoperability, and portability C++ code does have. On the contrary, the use of the C# language would have reduced the time needed to develop SUED, as this language comes along a wide framework that simplifies the developer's job (including network handling, XML parsing, etc) but it would have hindered its flexibility and would have been harder to couple SUED with OSG (which is a C++-based state-of-the-art graphics library). Nevertheless, the main reason to use C++ is the need to end up with an optimized code able to be used to build DVEs and suitable to be extended. As C++ is both the most used language and generates faster executables than managed languages, we think it is the right choice.

3.3 SUED Description

The presence of several network choices offers the possibility of implementing different synchronization strategies that meet the requirements of the actual DVE being deployed. Depending on the awareness requirements the DVE needs to satisfy, one can choose from the most restrictive (and easier to synchronize) master/slave which can provide frame-to-frame synchronization to the least restrictive (but most flexible) P2P model where an avatar could be slightly misplaced respect to its supposedly accurate location.

From the point of view of an application developer, a collaborative environment in SUED is expressed as a set of finite-state machines, running in each node, that are synchronized to each other. The description of the initial state machines is obtained from the XML configuration files by the "*Exercise*" module. The execution of the simulation in SUED relies on an event-based approach where each action performed on the simulated environment is introduced as an event and stored in the *'EventRepository'*. This repository contains actions performed by the local user (through the UserInterfaces component which process local machine interfaces and stores its actions in the repository) as well as the external actions received from remote users of the DVE system. Since the repository information is time-stamped, the "*DebriefingControl*" module allows users to record and reproduce the current simulation in real-time. This feature is very useful when DVE systems are used to train professionals on emergency plans, security procedures or even medical procedures.

The decision to redirect all user interfaces actions to the same repository where the state-machine events are stored (and extracted from to be executed), rather than keeping a separate repository, was taken to ensure that mutual interaction between the two is not only feasible but easy to deploy.

Furthermore, SUED provides the user with a simple mechanism that allows both the state-machine and the event repository to be dynamically modified by the execution of some special events (that we call meta-events because they don't refer to the simulation but to the description of the exercise itself) so the simulation behavior could be "reprogrammed". This reprogramming capabilities could be triggered both from a user interface and from the exercise itself, being the later much more common than the former.

This meta-programming ranges from delaying an event by an specific amount of time, to the deletion/modification of a transition or even a whole state. As any other integral part of SUED, more meta-events could be easily added if they were needed.

The "*StereoDisplayAndCameras*" module, or SDC, groups all the visualization and interaction capabilities into an easy-to-use interface that can be controlled with the XML configuration files. This module improves some of the capabilities included in OSG and OpenAL in order to produce the graphical and sound experience. The SDC module includes an improved high quality *off-axis* stereoscopy, 3D sound capabilities integrated into an OSG scenegraph, an advanced camera control and, even, a multimodal interface support including a low-cost optical tracking system. Figure 2 shows the parallel eye stereo projection included by default in SUED. In this type of stereo visualization the image on either side is lined up side-by-side, reducing, thus, the feeling of discomfort.

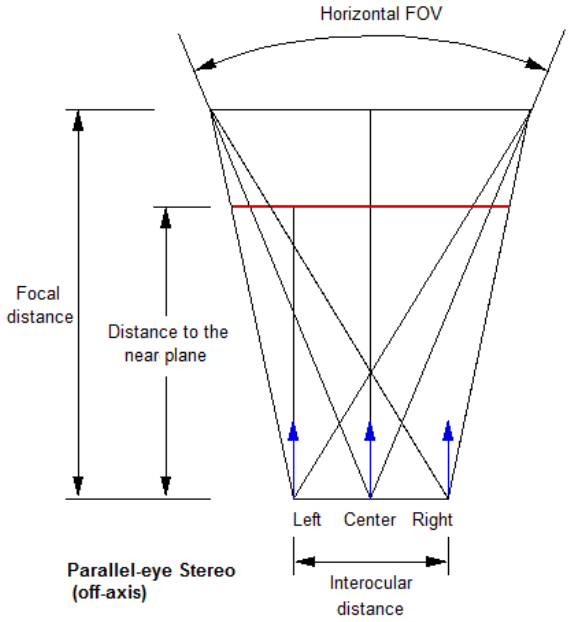

Figure 2: The stereo visualization method, based on a parallel eye projection, included by default in SUED

The addition of more user interfaces (such as steering wheels, multi-touch systems, motion capture) is not only encouraged and extremely simple but also being tested at some of our recent projects with very good results.

3.4 Using SUED

The following code shows a typical structure of an XML document where a simple simulation is modeled as a finite-state machine in SUED. This configuration file describes one of the different clients involved in an collaborative simulator of dumpers in construction sites. A dumper is a four-wheeled open vehicle used to move bulk material on industrial job sites or for smaller construction and landscaping jobs.

```
<?xml version="1.0" encoding="iso-8859-1" ?>

<Simulation ParticipantId="1" ShowAxes="false"
AmbientSound="../sounds/background.wav">

<DisplayProperties ShowCursor="false" Stereo="true"
DisplayMode="Normal" ScreenWidth="1.0" ScreenHeight="1.5"
UserToScreenDistance="1.0" />

  <!-- Scene definitions -->
  <Scene FileName="../models/Scene.ive" X="5.0" Y="10.0" />

  <!-- Multimodal interface definitions -->
  <UserInterfaces>
    <UserInterface Type="KeyboardNavigation" Name="Keyboard" SystemName="" />
    <UserInterface Type="MouseOrientation" Name="Mouse" SystemName="" />
```

```
    <UserInterface Type="Optitrack" Name="HeadTracker" SystemName="Head" />
    <UserInterface Type="SteeringWheel" Name="Wheel"
    SystemName="Logitech G25 Racing Wheel USB" />
  </UserInterfaces>

  <!-- Object definitions -->
  <Include File="../cfg/Dumper.Objects.xml" />

  <!-- Camera definitions -->
  <CameraList>
    <Camera Collidable="true" Name="FreeCamera" X="1.0" Y="0.0" Z="1.8"
    EyeSeparation="0.03" FocalDistance="20.0" Near="0.05" Far="5000.0" />
    <Camera Collidable="false" Name="DriverCamera" Z="1.5"
    ParentObject="Chasis" EyeSeparation="0.025" FocalDistance="15.0"
    Near="0.05" Far="5000.0" />
  </CameraList>

  <!-- Exercise definitions -->
  <Exercise Name="Dumper">
    <StateMachine Initial="1">
      <State Id="1" Name="Init">
        <IncomingEventList>
          <Event Participant="0" Type="SetFlag" Name="SeatBelt" Value="0" />
          <Event Participant="0" Type="SetFlag" Name="EmergencyLight" Value="0" />
          <Event Participant="-1" Type="ShowText" Text="Exercise initialized" />
        </IncomingEventList>

        <TransitionList>
          <Transition Destination="2" />
        </TransitionList>

        <OutgoingEventList>
          <Event Participant="0" Type="ObjectControlAnimations"
          Object="Chasis" Action="Play" />
          <Event Participant="0" Type="ObjectControlSound"
          Object="Chasis" Action="Play" />
        </OutgoingEventList>
      </State>

      <State Id="2" Name="Waiting">
        <IncomingEventList>
          <Event Participant="-1" Type="Text"
          Text="Waiting for your co-worker to pick the keys" />
          <Event Participant="-1" Type="Timer" Timer="Timer01"
          Command="Restart" />
        </IncomingEventList>

        <Transition Destination="4">
              <ConditionList UnionMode="And">
                <Condition Type="Flag" Name="EmergencyLight"
                TestType="Equal" TestValue="0" />
                <Condition Type="Timer" Timer="Timer01"
                TestType="Greater" TestValue="2.5" />
              </ConditionList>
          <EventList>
            <Event Participant="0" Type="ObjectControlGraphic"
            Object="ErrorHighlight" Command="Unhide" TimeStampOffset="2" />
          </EventList>
        </Transition>

        <TransitionList>
          <Transition Destination="3">
            <ConditionList>
              <Condition Participant="2" Type="ObjectPicked" Object="Keys" />
            </ConditionList>
          </Transition>
        </TransitionList>
      </State>

      <State Id="3" Name="Accessing">
        <IncomingEventList>
          <Event Participant="-1" Type="Text" Text="Hoop on" />
        </IncomingEventList>

        <TransitionList>
          <Transition Destination="5">
            <ConditionList>
              <Condition Type="ObjectWatched"
              Object="Chasis" Threshold="5.0" Far="5.0" />
            </ConditionList>
          </Transition>
        </TransitionList>
      </State>

      <State Id="4" Name="Warning1">
        <IncomingEventList>
          <Event Participant="-1" Type="Text"
          Text="Be sure to pick up the keys and switch on the emergency light" />
        </IncomingEventList>
      </State>

      <State Id="5" Name="Procedure">
        ...
        ...
      </State>

      ...
      ...
      ...

    </StateMachine>
  </Exercise>

  <Current Object="" Camera="FreeCamera" Exercise="Dumper" />
</Simulation>
```

SUED models each simulation as a set of seven component, which are declared within the XML configuration files. The component "*Simulation*" declares the unique identifier (ParticipantId) used to reference a given client in the DVE system. The component "*Scene*" enumerates the 3D model files which describe the 3D virtual environment where the driver can move an operate.

This scene may also have cameras integrated/defined within it. In fact the module is designed to directly import models built with Autodesk 3d Studio Max to speed up the designer's integration process. This way, the user does not need to explicitly define the hierarchy of the objects within the XML files, as it is read from the 3D model.

The component "*UserInterfaces*" declares all the definitions of the multimodal interfaces available for this client in the DVE system. In this example, the user has at one's disposal a keyboard, a mouse and a steering wheel to move the dumper within the virtual 3D construction site. Also, this user has an optional head tracking system to sense the position and direction of the viewer's head and to produce the correct stereo perspective. The component "*Include File*" allows software designers to incorporate XML files inside of some others XML files, and defines large simulations as a hierarchical structure of configuration files. The component "*CameraList*" defines a set of direct real-time virtual cameras within the virtual scene. These cameras remain identified and can be changed as the simulation advances.

It is also possible to define a series of "flags" that will remain available at SUED's user disposal to speed up the construction of the XML files representing the exercise.

Finally, the component "*Exercise*" describes the simulation of a set of procedures which, in turn, can be modeled as one or a few finite-state machines (StateMachine). Each finite state machine is defined by a number of states which are labeled (Name) and univocally identified (Id). The states define actions which could be launched when the simulation access to the state, or when it leaves to different states by means of predefined transitions. When an action is launched by a client (show a text, move an object, change a camera position, etc.) a new event, along with a time offset, is added into the event repository of that client. In this sense, the action dispatching can be executed instantaneously or deferred by means of programmed timers depending on the value of this offset. Moreover, and since all the operations related to the event repository are time-stamped with the granularity of milliseconds, SUED offers a debriefing mechanism to reproduce the simulation sometime later after it finishes.

As well as the features included in the SDC module, SUED also offers object positioning and new interfaces for animation control. In order to improve the framework, all of the functionalities provided by SUED correspond to a particular C++ class that can by extended by users using hierarchy mechanisms and through virtual operations.

4 Experiments and Applications

We have used SUED to develop two different DVE systems called *Previsor* and *Ensisam-II*, in order to validate our software framework. Previsor is a collaborative training tool to prevent from height risks on construction sites. Although this system is not oriented to support a high number of users connected simultaneously, it extensively exploits the multimodal paradigm and the state-machine-based procedural module. On the other hand, Ensisam-II is a collaborative simulation environment designed to instruct and evaluate evacuation procedures on marine rescue emergencies. It is a fully immersive DVE system oriented to support hundreds of simultaneous clients.

In order to establish a fair comparison among the different development alternatives, these DVE systems were developed in parallel by three different development teams, randomly chosen for each DVE system from trained members of our organization. Each development team was composed by an analyst programmer, a 3D

Figure 3: Some snapshots of the Previsor (left) and Ensisam-II (right) DVE systems when they were developed on SUED

designer and three full-time experienced programmers. The first development team followed a classic development of DVE systems using open-source and well-known VR software tools such as OSG, CAL3D, etc. On the contrary, the second and third teams followed a development based on frameworks for DVE systems using Delta3D[5] and SUED, respectively.

As expected, the final visual aspect and functionality of the DVE system developed by the different teams were very similar. From the point of view of the system performance, measured in frames per second (FPS), the results did not shows significant differences related to the development framework. Figure 3 shows some snapshots obtained from the execution of different simulation procedures in Previsor and Ensisam-II.

Table 1 shows the result of measuring some parameters related to the development time and the number of lines of code produced by the three different teams working on the same DVE system. This table shows the total number of source lines of code included within the final DVE system as well as the actual number of lines of code written by the programmers. We have denoted these parameters as *SLOC* and *DLOC*, respectively. Moreover, this table shows the development time (expressed in weeks) of each implementation of the DVE systems performed by the different teams. We have denoted these parameters as *TT*, which has been divided into two different periods corresponding to the coding stage (*CT*) and the debugging stage (*DT*) of the software projects.

The results shown in Table 1 seem to indicate that the final sizes of the developed DVE systems get higher when using a non-classic approach. Nevertheless, the number of new lines of source code the programmers really need to create (DLOC) gets lower when using SUED or Delta3D. In addition, the total time (TT) needed by the third development team was the shortest in both DVE systems. These results expose that the time invested deploying SUED is amortized as it gets easier and faster to deploy a new system.

As the Table 1 shows the use of SUED reduced the time needed for the deployment of the system and shortened the project duration by roughly 35% in Previsor and 25% in Ensisam-II, when compared to the first development team. The results represent a reduction of 20% in both DVE systems, when compared to the development based on Delta3D.

The main reason for these time differences is related to the use of the configuration mechanism included in SUED. This module allows speeding up the debugging stage of the software (reducing the DT value) and reducing the duration of the development of a given DVE system. The configuration mechanism, based on XML files, allows developers to reduce the time needed to perform the tests required to validate the system since errors and changes do not need to be recompiled. This time and cost reduction, not only achieved in the development phase of the project, are due to the possibility of using XML files in the evaluation and tuning of the system as well as in the final testing and deployment of the simulation procedures.

Table 1: Parameters related to the development time and project size of the performed implementations

Obtained Results			
Dev.Team	Parameter	Ensisam-II	Previsor
Team 1	SLOC	84650	93140
	DLOC	8450	10620
	CT (weeks)	27	29
	TT (weeks)	12	13
	TT (weeks)	39	42
Team 2	SLOC	102400	107650
	DLOC	6030	7260
	CT (weeks)	23	25
	DT (weeks)	8	9
	TT (weeks)	31	34
Team 3	SLOC	96520	101250
	DLOC	5890	7325
	CT (weeks)	22	20
	DT (weeks)	4	4
	TT (weeks)	25	24

5 CONCLUSIONS

In this paper, we have proposed SUED, a new open-source software framework designed to enable a rapid prototyping of low-cost DVE systems. Unlike other proposals, SUED simplifies the process of developing complex DVE systems by means of an easy-to-use interface based on XML files. This configuration system tries to minimize the time needed to modify and adapt the platform source code for the development of a given DVE system. Moreover, the standard capabilities of SUED can be easily extended by defining custom modules on open-source and well-known VR libraries.

In order to validate our framework, two different case studies based on actual DVE systems were developed. The experiments performed using SUED showed a significant reduction of the scheduled project time when compared to classic developments or even to other cutting-edge development frameworks.

6 FUTURE WORK

As future work, we plan to add more features to the kernel module including fluid simulation, motion capture and dynamic control. These features need currently to be added when deploying a new simulation, and we intend to integrate them into SUED's core.

Also, we intend to integrate a new network protocol based on multicast transmission to the networking module. In addition, it is also planned to add and test a P2P overlay-based network protocol, in order to simplify and automatize the management of the network.

And last but not least, it will be very useful to be able to create the exercises with a visual tool, so to build some sort of IDE would be a good idea in order to speed up the process of building new simulations.

REFERENCES

[1] B. Delaney. *The Market for Visual Simulation/Virtual Reality Systems*. CyberEdge Information Services, seventh edition edition, 2004.

[2] C. Endres, A. Butz, and A. MacWilliams. A Survey of Software Infrastructures and Frameworks for Ubiquitous Computing. *Mobile Information Systems Journal, IOSPress*, 1:41–80, January 2005.

[3] S. Kim, D. Gracanin, W. Winchester, and T. Kuc. The AR-CAVE: Distributed Collaborative Augmented Reality and immersive virtual reality system. In *Proceedings of the 3rd International Conference on Autonomous Robots and Agents (ICARA 2006)*, pages 351–354, Palmerston North, New Zealand, December 2006.

[4] J. Ohlenburg, I. Herbst, I. Lindt, T. Frhlich, and W. Broll. The MORGAN Framework: Enabling Multi-User AR and VR Projects. In *Proceedings of the Symposium on VR Software and Technology (VRST)*, pages 166–169, Hong Kong, November 2004. ACM Press.

[5] E. J. R. Darken, P. McDowell. Projects in VR: the Delta3d open source game engine. *IEEE Computer Graphics and Applications, IEEE Computer Society*, 25:10–12, May-June 2005.

[6] T. Reicher, A. MacWilliams, B. Bruegge, and G. Klinker. Results of a Study on Software Architectures for Augmented Reality Systems. In *Proceedings of the 2nd IEEE and ACM International Symposium on Mixed and Augmented Reality (STARS)*, page 274, Tokio, Japan, October 2003. IEEE Computer Society.

[7] L. Renambot, H. Bal, D. Germans, and H. Spoelder. CAVEStudy: An Infrastructure for Computational Steering and Measuring in Virtual Reality Environments. *Cluster Computing*, 4:79–87, March 2001.

[8] R. Salvatore. Using Open Source Software in Visual Simulation Development. Master's thesis, Naval Postgraduate School Monterey CA, Department of Computer Science, September 2005.

[9] S. Singhal and M. Zyda. *Networked Virtual Environments: Design and Implementation*. Addison-Wesley, 1999.

[10] O. Staadt, M. Nf, E. Lamboray, and S. Wrmlin. Jape: A Prototyping System for Collaborative Virtual Environments. In *Proceedings of the ACM Eurographics 2001*, pages 8–16, Manchester, UK, September 2001. ACM Press.

[11] H. Tramberend. Avocado: A Distributed Virtual Environment Framework. In *Proceedings of the IEEE Virtual Reality Conference (VR'99)*, pages 14–21, Houston, USA, March 1999. IEEE Computer Society.