



Simulating Collision Avoidance by a Reactive Agent Using VRML

by Andrew M. Neiderer

ARL-TR-3566

August 2005

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory

Aberdeen Proving Ground, MD 21005-5067

ARL-TR-3566

August 2005

Simulating Collision Avoidance by a Reactive Agent Using VRML

Andrew M. Neiderer

Computational and Information Sciences Directorate, ARL

REPORT DOCUMENTATION PAGE				<i>Form Approved</i> <i>OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.					
1. REPORT DATE (DD-MM-YYYY) August 2005		2. REPORT TYPE Final		3. DATES COVERED (From - To) December 2004–February 2005	
4. TITLE AND SUBTITLE Simulating Collision Avoidance by a Reactive Agent Using VRML				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Andrew M. Neiderer				5d. PROJECT NUMBER P622783.Y10	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) U.S. Army Research Laboratory AMSRD-ARL-CI-CT Aberdeen Proving Ground, MD 21005-5067				8. PERFORMING ORGANIZATION REPORT NUMBER ARL-TR-3566	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT Prototype nodes using the Virtual Reality Modeling Language (VRML) 2.0 standard have been developed for reactive, agent-based route navigation within a three-dimensional synthetic environment. The agent has limited intelligence with the basic constraint of noncollision by enforcing some minimal distance of approach to objects: this includes both static and moving obstacles. Navigation is nontrivial since the location of the goal may be changing and the agent reacts accordingly. An existing VRML model of a building was imported, and the subsequent exposedField interface for repulsion was defined. Simulation of an agent within this context was then verified. An open source development using VRML encourages a “think small, scale large” philosophy for a network of objects.					
15. SUBJECT TERMS collision avoidance, web-based visualization, interactive virtual reality					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT UL	18. NUMBER OF PAGES 30	19a. NAME OF RESPONSIBLE PERSON Andrew M. Neiderer
a. REPORT UNCLASSIFIED	b. ABSTRACT UNCLASSIFIED	c. THIS PAGE UNCLASSIFIED			19b. TELEPHONE NUMBER (Include area code) 410-278-3203

Contents

List of Figures	iv
1. Introduction	1
2. VRML Basics	2
3. VRML Implementation	4
4. Conclusion and Future Efforts	5
Appendix. VRML Prototypes for Collision Avoidance Example	6
Distribution List	23

List of Figures

Figure 1. An abbreviated list of VRML nodes.	3
Figure A-1. Animation snapshot.....	8

1. Introduction

Urban operations, such as responding to a medical emergency or even a rescue operation involving hostages, continue to increase as the world's population congregates in central locations. Rehearsal for these types of missions results in increased precision and speed. A concern of the Tactical Collaboration and Data Fusion Branch at the U.S. Army Research Laboratory (ARL) is preparing soldiers for military operations on urban terrain (MOUT) where timing is critical for success. Recently we developed software that not only has application for MOUT, but could also be used in simulation of collision avoidance in military operations other than war.

A military rescue team, which usually includes dismounted infantry, needs to reach their target as fast as possible. In the following discussion the term "agent" is used in the context of an infantry soldier. The metric we chose for path selection is minimizing the distance from an agent to the target with the basic constraint of noncollision. Collision avoidance is simulated by defining a repulsion exposedField of a Virtual Reality Modeling Language (VRML) prototype (PROTO) node for a particular object within the virtual environment (VE). A general restriction is that an agent must obey the general integrity rule of gravity and cannot "step over" an obstacle. There are other concerns of an agent such as minimizing exposure or minimal risk. The optimal solution for a particular situation may be a combination of the above, or others. But for now the maximum reward for a particular simulation is representative of straight path dashes over short distances, which is typical of a dense urban environment. We also assume that the agent knows the exact location of the target at all times, as if he sees it.

Operations in an urban environment can also be quite chaotic, and adjustments may be necessary. In all cases, the agent should be reactive to both static and dynamic events that are encountered along its selected path. For example, an existing structure may undergo an impact from a munition either accidentally or intentionally for strategic reasons. A soldier may then need to alter his route based on the resultant rubble, or possibly he observes an enemy patrol that needs to be avoided. There is little time to analyze data in most circumstances, and often decisions are based on reactions. Therefore, some deviation from the original plan will likely occur, but our software agent always tries to minimize distance to the target from the current location.

Note that an additional challenge in real-time simulation within an urban setting is the approximation of complex phenomena such as the previously mentioned. We have developed a physics-based algorithm for real-time display of munitions penetrating urban structures.¹ We

¹ Neiderer, A. M.; Thomas, M. A.; Pearson, R. *A Fracturing of Polygonal Objects*; ARL-TR-1649; U.S. Army Research Laboratory: Aberdeen Proving Ground, MD, April 1998.

then developed software for the distribution of fragments resulting from such fractures.² Maintaining a real-time response in our simulation work is a top priority.

This report presents an approach for collision avoidance by a reactive agent within a VE. First, we provide an introduction to version 2.0 of VRML, hereafter simply referred to as VRML. Its successor, the Extensible Three-Dimensional (X3D) modeling language, is an XML-encoding of VRML that is being reviewed by the International Standards Organization for acceptance; it has reached final draft but has not been officially standardized as of November 2004. X3D has some additional features and may be a better solution, but we will wait and see if this language becomes widespread as has VRML. Section 3 provides details of our application, with full source code given in the appendix. It runs on both Microsoft Windows XP workstations using the Cortona VRML client by ParallelGraphics, Inc. (<http://www.parallelgraphics.com/products/cortona/>) or the VRML plug-in from Octaga, Inc. (<http://www.octaga.com/>), and Red Hat Enterprise Linux microcomputers using the standalone VRML application FreeWRL (<http://www.crc.ca/FreeWRL/>). Note that there is a VRML standalone for a personal digital assistant (PDA), but we have not evaluated it (<http://www.alphaworks.ibm.com/tech/mobile3d>). Finally, we conclude and suggest where further work should be done to make this a better application.

2. VRML Basics

VRML is a Web-based technology for describing and delivering a three-dimensional (3-D) interactive VE over the Internet. The standard defines over 60 primitives, called nodes, with a capability of extension through user-defined PROTO nodes. A scene graph description of a VE is a hierarchical organization of the possible nodes (see figure 1):

1. Shape node(s) for both a simple geometry and a general geometry, which is built using actual coordinate values, including its appearance,
2. Light source node(s) to control lighting within the virtual world,
3. Grouping node(s) for allowing a collection of nodes to be defined and manipulated as one,
4. Sensor and interpolator nodes to allow for animation, and
5. Script node(s) for passing information between nodes by events along a defined ROUTE.

² Neiderer, A. M.; Thomas, M. A.; Hansen, C. E. *Distribution of Fragments Resulting From Polygonal Object Fracture*; ARL-TN-182; U.S. Army Research Laboratory: Aberdeen Proving Ground, MD, September 2001.

<i>Shape Node</i>
Geometry Nodes
<i>Box</i>
<i>Sphere</i>
<i>Cone</i>
<i>Cylinder</i>
<i>IndexedLineSet</i>
<i>IndexedFaceSet</i>
<i>Extrusion</i>
<i>ElevationGrid</i>
<i>Text</i>
<i>Appearance Node</i>
Light Source Nodes
<i>DirectionalLight</i>
<i>PointLight</i>
Grouping Nodes
<i>Group</i>
<i>Transform</i>
<i>Collision</i>
<i>Switch</i>
<i>LOD</i>
Sensor Nodes
<i>TouchSensor</i>
<i>ProximitySensor</i>
<i>PlaneSensor</i>
<i>TimeSensor</i>
Interpolator Nodes
<i>OrientationInterpolator</i>
<i>PositionInterpolator</i>
<i>Script Node</i>

Figure 1. An abbreviated list of VRML nodes.

VRML supports animation and user interaction of a 3-D VE through an event-driven model. Every node has a field interface (field, exposedField) and an event interface (eventIn, eventOut). A scene graph is made dynamic by receiving and sending events through the node event interface. A ROUTE definition is used to specify the flow of events between nodes. ROUTE wires an output event (eventOut) of a sensor node to the input event (eventIn) of an interpolator

node, and then another ROUTE from this interpolator node to a transform node. In addition, script nodes can be associated with Java or JavaScript for treating and modifying events.

Although VRML offers a compelling presentation of 3-D data and provides for an interactive scene graph, it was not designed to explicitly support bi-directional network communication. There are some roundabout ways of “pushing” and “pulling” data from the network. One example is dynamic Web content created by using HTTP communication with Web services such as JavaServer Pages technology.

Two networking improvements have been added to the proposed X3D standard to address the above VRML shortcoming for a network of nodes: (1) the addition of a load field to the Inline node so that an application can better control when assets are loaded, and (2) the definition of a LoadSensor node so that an application can know when assets are loaded in order to be more responsive.

When X3D becomes fully standardized and browser plug-ins become more widespread and supported, we will re-evaluate our application in that context. It is even likely that VRML-to-X3D translator(s) will be available to ease the transition if we choose to do so.

3. VRML Implementation

If a collision node(s) is not specified for a VRML scene, then the browser is responsible for detecting collision of an agent with objects in the VE during navigation. The collision node is a grouping node in the scene graph. Any geometric primitives within the group, i.e., MFNode children of the exposedField interface, are collide-able assuming that the exposedField collide is set to true. Because we wanted to avoid collision, not merely detect it, we created PROTOs and used a VRML node developed by Peter Gerstmann³ at The Ohio State University: AvoidObstacles.

The PROTO AvoidObstacles defines the initial position of the agent (exposedField position) and the target (field goal); both are 3-D floating point vectors, i.e., SFVec3f. Also recall that both of these fields are dynamic, and can change values over time as the agent reacts but continues to avoid any obstacles along its path. All obstacles are also PROTOs and must have an initial position and a repulsion value; the exposedField position is a 3D floating point vector; and the larger the floating point value for repulsion, the larger the influence of the circular field.

Complete VRML code is given in the attached appendix. The main source AvoidBuildingsAndObjectsExample.wrl includes the external prototype (EXTERNPROTO)

³ Gerstmann, P.; *Building Games in VRML*; The College of The Arts, The Ohio State University, May 2000.

declarations for the above nodes. Each additional obstacle must be declared and defined as mentioned, and then added to the MFNode obstacles field of AvoidObstacles.

4. Conclusion and Future Efforts

ARL has developed VRML PROTO nodes for simulating route navigation of minimal distance for a reactive agent within a VE consisting of both static and dynamic obstacles. The next step is simulation within a more detailed and populated VE, representative of an urban environment, using additional VRML nodes. We believe that this will be relatively straightforward since we chose a language that is a standard for the representation of VR environments.

The Scenario Authoring and Visualization for Advanced Graphical Environment (SAVAGE) project at the Naval Postgraduate School (<http://web.nps.navy.mil/~brutzman/Savage/>) in Monterey, CA, has a rather large collection of military models developed in VRML/X3D. Another archive of VRML nodes for urban model representation that are freely available exists at <http://www.int3d.com/>. However, a VE consisting of N moving objects and M stationary objects involves keeping track of $\binom{N}{2} + NM$ pairs of objects for each time interval.⁴ Therefore, we must research intersection techniques further to reduce this computation if we want to maintain interactive frame rates.

Another enhancement of this work would be the replacement of VRML primitive nodes for the data definition of an agent. A possibility is a soldier representation of the agent, e.g., using Discreet's Character Studio, with motion-capture technology for movement within the VE. Motion capture data typical of nuances of a soldier will include walking, running, taking cover, rising from the ground, and communicating through hand signaling. We may then attempt to replace the reactive capability of our agent in route navigation with some further intelligence, perhaps based on cognitive maps.

⁴ Badawy, W. A.; Kelash, H. M. Collision Detection in VRML: A Survey; *Emirates Journal for Engineering Research* 2003, 8 (2).

Appendix. VRML Prototypes for Collision Avoidance Example

The following VRML code includes prototypes that define a primitive structure (ObstacleBldg.wrl) and character (ObstacleGuard.wrl) to demonstrate collision avoidance of an agent, which is described using primitive geometry also. The intent here is to define only single static and a dynamic nodes within the VE, keeping it clutter-free for the agent so that we can visually verify the VRML model. We now plan on adding more VRML nodes to approach reality.

A.1 ObstacleBldg.wrl

```
#VRML V2.0 utf8

## ObstacleBldg.wrl:  ObstacleBldg node
##
## description:        obstacle prototype of a building
##
## author:             Andrew M. Neiderer
##                    23 September 2004

PROTO ObstacleBldg [

    # field interface

    field          SFVec3f size          4.0 4.0 4.0

    # exposedField interface

    exposedField SFVec3f position  0.0 0.0 0.0
    exposedField SFFloat repulsion 4.0
]

# definition

{
    Transform {
        translation IS position
        children [
            Inline {
                url
                "http://web.nps.navy.mil/~brutzman/Savage/Buildings/ErdcTwoStoryBuilding/ErdcTwoStoryBuilding.wrl"
            }
            # url "ErdcTwoStoryBuilding.wrl"
            # url "building_1.wrl"
        ]
    }
}
```

A.2 ObstacleGuard.wrl

```
#VRML V2.0 utf8

## ObstacleGuard.wrl:  ObstacleGuard node
##
## description:          obstacle prototype of a guard
##
## author:               Andrew M. Neiderer
##                      3 September 2004

PROTO ObstacleGuard [

    # field interface

    field          SFFloat radius      1.0

    # exposedField interface

    exposedField SFVec3f position  4.0 0.0 7.5
    exposedField SFFloat repulsion 4.0
]

# definition

{
    Transform {
        translation IS position
        children [
            Shape {
                appearance DEF RED Appearance {
                    material Material {
                        diffuseColor 1.0 0.0 0.0
                    }
                }
                geometry Cylinder {
                    height 1.0 radius 0.25
                }
            }
            Transform {
                translation 0.0 0.7 0.0
                children [
                    Shape {
                        appearance USE RED
                        geometry Sphere {
                            radius 0.20
                        }
                    }
                ]
            }
        ]
    }
}
```

A snapshot of the animation is given in figure A-1. The target is located inside the virtual building. The blue figure, which is our software agent, tries to reach it as fast as possible by entering through the doorway and avoiding all obstacles that are encountered along its route. The red figure is representative of a guard patrolling the entrance of the building, moving west to east and then reversing for the entire width of the building; the initial position and repulsion are declared in the exposedField interface of the ObstacleGuard.wrl. The building is likewise an obstacle for the agent with the exposedField interface given in ObstacleBldg.wrl. All actual values are defined in AvoidBuildingsAndObjectsExample.wrl, which is the main VRML file. The agent and its starting position are defined here, as well as any additional animation within the VE.

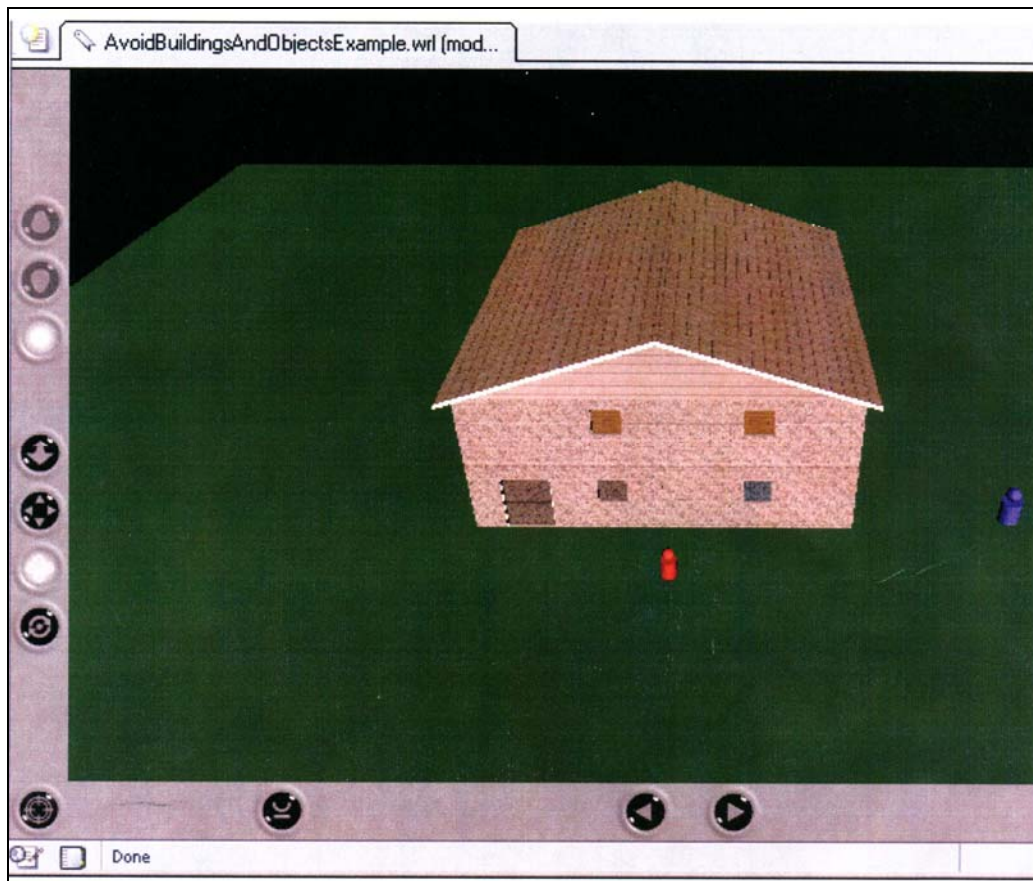


Figure A-1. Animation snapshot.

A.3 AvoidBuildingsAndObjectsExample.wrl

```
#VRML V2.0 utf8
```

```
## AvoidBuildingsAndObjectsExample.wrl
```

```
##
```

```
## description:
```

```
##
```

```
## author:      Andrew M. Neiderer
```

```
##              23 September 04
```

```

# xxx

EXTERNPROTO AvoidObstacles [

    # field interface

    field      SFVec3f    goal
    field      SFFloat    goalPriority
    field      SFFloat    speed
    field      SFString    state
    field      MFNode      obstacles

    # exposedField interface

    exposedField SFVec3f    position
    exposedField MFNode      children

    # event interface

    eventIn      SFTIME    updateRequest
    eventIn      SFVec3f    set_goal
    eventIn      SFString    set_state
    eventIn      MFNode      set_obstacles

] "AvoidObstacles.wrl"

# a building

EXTERNPROTO ObstacleBldg [

    # field interface

    field      SFVec3f    size

    # exposedField interface

    exposedField SFVec3f    position
    exposedField SFFloat    repulsion

] "ObstacleBldg.wrl#ObstacleBldg"

# a guard

EXTERNPROTO ObstacleGuard [

    # field interface

    field      SFFloat    radius

    # exposedField interface

    exposedField SFVec3f    position
    exposedField SFFloat    repulsion

] "ObstacleGuard.wrl#ObstacleGuard"

```

```

# xxx

EXTERNPROTO MetaData [

    # exposedField interface

    exposedField MFString instructions
    exposedField MFString name
    exposedField MFString description
    exposedField MFString interface

] "MetaData.wrl"

MetaData{
    name ["MetaData"]
    description [
        "descriptive info display for protos."
        "typically, protos present the user with an empty screen;"
        "by using a MetaData proto, you can provide helpful info"
        "and a printable interface."
    ]
    interface [
        "EXTERNPROTO MetaData ["
        "    exposedField MFString instructions "
        "    exposedField MFString name "
        "    exposedField MFString description "
        "    exposedField MFString interface "
        "]" \"MetaDataPROTO.wrl\"
    ]
}

# scene setup

Viewpoint {
    description "bird's eye view"
    position 0.0 33.5 0.027
    orientation -1.0 0.0 0.0 1.57
    jump FALSE
}

Viewpoint {
    description "on the level"
    position 0.0 19.388 27.32
    orientation -1.0 0.0 0.0 0.617
    jump FALSE
}

NavigationInfo {
    type "EXAMINE"
}

WorldInfo {
    info [ "File produced by IMSI FloorPlan V4" ]
    title "FP2StoryTextures.wrl"
}

NavigationInfo {

```

```

    type [ "EXAMINE" "ANY" ]
}

DEF VP_Z_POS Viewpoint {
    description "Front"
    position -1.8288 1.5 19.848103
}

DEF VP_Z_NEG Viewpoint {
    description "Back"
    orientation 0.0 1.0 0.0 3.141592
    position -1.8288 1.5 -20.457703
}

Viewpoint {
    description "View 1"
    position -4.8768 1.5 2.7432
}

Viewpoint {
    description "View 2"
    position -1.8288 1.5 -2.3622
}

Viewpoint {
    description "View 3"
    position 1.2192 1.5 2.7432
}

Viewpoint {
    description "View 4"
    position -4.8768 1.5 2.7432
}

Viewpoint {
    description "View 5"
    position -1.8288 1.5 -2.3622
}

Viewpoint {
    description "View 6"
    position 1.2192 1.5 2.7432
}

# obstacles

DEF 01 ObstacleBldg {
    position 0.0 0.0 0.0
    repulsion 675.0
}

DEF 02 ObstacleGuard {
    radius 1.0
    position 9.0 0.0 0.0
    repulsion 25.0
}

```

```

# for animation

DEF PI PositionInterpolator {
  key [0.0 0.33 0.66 1.0]
  keyValue [3.0 0.15 8.0, 5.0 0.15 8.0, -8.0 0.15 8.0, 3.0 0.15 8.0]
}

DEF TS TimeSensor {
  cycleInterval 10
  loop TRUE
}

# target

Transform {
  rotation 1.0 0.0 0.0 -1.57
  children [
    DEF GOAL Transform {

#    position of target is outside the building
#    translation -15.0 -0.0 1.0

#    position of target is inside the building
translation -4.5 -4.75 0.0

children [
  Transform {
    translation 0.0 0.0 -0.25

# scale, ie size of, target
scale 1.5 1.5 1.0

children [
  DEF CORNER Shape {
    geometry IndexedLineSet {
      coord Coordinate {
        point [-0.5 -0.25 0.0, -0.5 -0.5 0.0, -0.25 -0.5 0.0]
      }
      coordIndex [0 1 2 -1]

# color of target
color Color {
  color [0.0 0.0 1.0]
}
colorIndex [0 0 0 -1]
}
}
Transform {
  rotation 0.0 0.0 1.0 1.57
  children [USE CORNER]
}
Transform {
  rotation 0.0 0.0 1.0 -1.57
  children [USE CORNER]
}
Transform {
  rotation 0.0 0.0 1.0 3.1415

```

```

        children [USE CORNER]
    }
}
]
}
]
}
]
}
]
}

```

floor

```

Transform {
    translation 0.0 -1.0 0.0
    rotation 1.0 0.0 0.0 -1.57
    children [
        DEF PS PlaneSensor {}
        Shape {
            appearance Appearance {
                material Material {
                    diffuseColor 0.4 0.4 0.4
                }
            }
            geometry Box {
                size 24.0 24.0 1.0
            }
        }
    ]
}

```

Script node

```

DEF S Script {
    field SFNode g USE GOAL
    field SFNode p USE PS
    field SFVec3f offset 0.0 0.0 0.0

    eventIn SFVec3f set_pos
    eventOut SFVec3f goal_changed

    url ["javascript:
        function set_pos(pos)
        {
            var mv = offset.add(pos);

            goal_changed = new SFVec3f(mv.x,0.0,-mv.y);
        }
        function initialize()
        {
            p.set_offset = g.translation;
            set_pos(g.translation);
        }
    "]
}

```

chaser

```

DEF A AvoidObstacles {

```

```

position 20.0 0.1 0.0
goal     -4.0 0.0 0.0
speed    3.0
obstacles [
    USE 01
    USE 02
]
children [
    Shape {
        appearance DEF BLUE Appearance {
            material Material {
                diffuseColor 0.0 0.0 1.0
            }
        }
        geometry Cylinder {
            height 1.5 radius 0.35
        }
    }
    Transform {
        translation 0.0 1.0 0.0
        children [
            Shape {
                appearance USE BLUE
                geometry Sphere {
                    radius 0.25
                }
            }
        ]
    }
]
}

# animation

ROUTE TS.time TO A.updateRequest
ROUTE PS.translation_changed TO GOAL.set_translation
ROUTE PS.translation_changed TO S.set_pos
ROUTE S.goal_changed TO A.set_goal

ROUTE TS.fraction_changed TO PI.set_fraction
ROUTE PI.value_changed TO O2.set_position
}

```

A.4 AvoidObstacles.wrl

```
#VRML V2.0 utf8
```

```

## AvoidObstacles.wrl: AvoidObstacles prototype
##
## description: at each update request, the children node[s] are moved closer
##              (if chasing), or farther (if not chasing) from goal, avoiding
##              the obstacles in the obstacle list.
##
## author:      peter gerstmann
##              pgerstma@cgrg.ohio-state.edu
##              http://www.cgrg.ohio-state.edu
##

```

```

## revised by: Andrew M. Neiderer
##             3 August 2004
##
## notes:      acknowledge Matt Lewis, whose Virtual Environments course
provided
##             the basis code for this PROTO.
##             http://www.cgrg.ohio-state.edu/~mlewis
##

PROTO AvoidObstacles [

  # field interface

  field      SFVec3f    goal 0.0 0.0 0.0
  field      SFFloat    goalPriority 2.0
  field      SFFloat    speed 1.0
  field      SFString   state "CHASING"
  field      MFNode     obstacles []      ## each obstacle must have
                                          ## exposedField SFVec3f position
                                          ## exposedField SFFloat

  repulsion

  # exposedField interface

  exposedField SFVec3f    position 0.0 0.0 0.0
  exposedField MFNode     children []

  # event interface

  eventIn      SFTIME    updateRequest
  eventIn      SFVec3f    set_goal
  eventIn      SFString   set_state
  eventIn      MFNode     set_obstacles
]

# definition

{
  Group {
    children [
      DEF MOVER Transform {
        translation IS position
        children IS children
      }
      DEF SCRIPT Script {
        directOutput TRUE

        field      SFVec3f    goal IS goal
        field      SFFloat    goalPriority IS goalPriority
        field      SFFloat    speed IS speed
        field      SFString   state IS state
        field      MFNode     obstacles IS obstacles
        field      SFNode     mv USE MOVER
        field      SFVec3f    direction 0.0 0.0 1.0
        field      SFTIME    lastBeat 0.0
        field      SFBool     first TRUE
        field      SFFloat    power 2.5
      }
    ]
  }
}

```

```

field    SFFloat    wanderInc 0.25

eventIn  SFTIME      update IS updateRequest
eventIn  SFVec3f      set_goal IS set_goal
eventIn  MFNode       set_obstacles IS set_obstacles
eventIn  SFString     set_state

url ["javascript:

function repulseForce(obstacle)
{
    var vec = mv.translation.subtract(obstacle.position);
    var ods = obstacle.repulsion * 1.0 / Math.pow(vec.length(),power);

    return (vec.normalize()).multiply(ods);
}

function calcDirection()
{
    var newDir;

    if ( state == 'WANDER' ) {
        var r = Math.floor(Math.random() * 3.0);

        // use internal var instead of goal, and set it to goal in other
states
        var g = goal;
        g[r] += wanderInc;
        set_goal(g);
    }

    // compute force of attraction towards goal
    var gForce =
((goal.subtract(mv.translation)).normalize()).multiply(goalPriority);

    if ( state == 'FLEE' ) {
        gForce = gForce.inverse();
    }

    // compute force of repulsion from obstacles

    var rForce = new SFVec3f(0.0,0.0,0.0);

    for ( i = 0; i < obstacles.length; i++ ) {
        rForce = rForce.add(repulseForce(obstacles[i]));
    }

    direction = (gForce.add(rForce)).normalize();
}

function set_goal(val)
{
    goal = val;
}

function set_obstacles(val)

```

```

    {
        obstacles = val;
    }

    function set_state(str)
    {
        state = str;
    }

    function update(val)
    {
        if ( first ) {
            first = FALSE;
        }
        else {
            var timeElapsed = val - lastBeat;

            calcDirection();
            mv.translation = mv.translation.add(direction.multiply(speed *
timeElapsed));
        }

        lastBeat = val;
    }

    " ]
    }
    ]
}

```

A.5. Building_1.wrl

#VRML V2.0 utf8

building_1.wrl: VRML description of building 1

##

author: Andrew M. Neiderer

3 September 2004

##

description:

##

notes:

```

##

Group {
  children [
    Shape {
      appearance Appearance {
        material Material {
          emissiveColor 1.0 0.0 0.0
          transparency 0.6
        }
      }
      geometry Box {
        size 8.0 2.0 4.0
      }
    }
  ]
}

```

A.6. MetaData.wrl

```
#VRML V2.0 utf8
```

```

## MetaData.wrl:  MetaData prototype
##
## author:         peter gerstmann
##                pgerstma@cgrg.ohio-state.edu
##                http://www.cgrg.ohio-state.edu
##

```

```
## description:    provides descriptive info for proto files
```

```
PROTO MetaData [
```

```
    # exposedField interface
```

```
    exposedField MFString instructions [
```

```
        " This is an externproto definition file."
```

```
        " To include, add the following in your vrml file:"
```

```
        "     (click text to print to vrml console,"
```

```
        "     for easy cut and paste)"
```

```
    ]
```

```
    exposedField MFString name []
```

```
    exposedField MFString description []
```

```
    exposedField MFString interface []
```

```
]
```

```
# definition
```

```
{
```

```
    Transform {
```

```
        translation -3.75 4.5 0.0
```

```
        children [
```

```
            Transform {
```

```
                translation 0.0 0.0 0.0
```

```
                children [
```

```
                    Shape {
```

```

    geometry Text {
        fontStyle DEF FS FontStyle {
            family ["SANS"]
            size 0.25
        }
        string IS instructions
    }
}
]
}

Transform {
    translation 0.0 -1.5 0.0
    children [
        Shape {
            geometry Text {
                fontStyle FontStyle {
                    family ["SANS"]
                    style "BOLD"
                    size 0.25
                }
                string IS name
            }
        }
    ]
}

Transform {
    translation 0.0 -2.0 0.0

```

```

children [
  Shape {
    geometry Text {
      fontStyle USE FS
      string IS description
    }
  }
]
}
Transform {
  translation 0.0 -3.5 0.0
  children [
    DEF TS TouchSensor {}
  Shape {
    geometry DEF TXT Text {
      fontStyle FontStyle {
        family ["TYPEWRITER"]
        size 0.3
      }
      string IS interface
    }
  }
]
}

```

```

DEF S Script {
  directOutput TRUE

```

```

field    SFNode txt USE TXT

eventIn SFBool print_isActive

url["javascript:
    function print_isActive(itIs)
    {
        if ( itIs ) {
            print(' ');
            for ( var i = 0; i < txt.string.length; i++ ) {
                print(txt.string[i]);
            }
        }
    }
    "]
}

]
}

ROUTE TS.isActive TO S.print_isActive
}

```

NO. OF
COPIES ORGANIZATION

1 DEFENSE TECHNICAL
(PDF INFORMATION CTR
ONLY) DTIC OCA
8725 JOHN J KINGMAN RD
STE 0944
FORT BELVOIR VA 22060-6218

1 US ARMY RSRCH DEV &
ENGRG CMD
SYSTEMS OF SYSTEMS
INTEGRATION
AMSRD SS T
6000 6TH ST STE 100
FORT BELVOIR VA 22060-5608

1 INST FOR ADVNCD TCHNLGY
THE UNIV OF TEXAS
AT AUSTIN
3925 W BRAKER LN STE 400
AUSTIN TX 78759-5316

1 DIRECTOR
US ARMY RESEARCH LAB
IMNE ALC IMS
2800 POWDER MILL RD
ADELPHI MD 20783-1197

3 DIRECTOR
US ARMY RESEARCH LAB
AMSRD ARL CI OK TL
2800 POWDER MILL RD
ADELPHI MD 20783-1197

3 DIRECTOR
US ARMY RESEARCH LAB
AMSRD ARL CS IS T
2800 POWDER MILL RD
ADELPHI MD 20783-1197

ABERDEEN PROVING GROUND

1 DIR USARL
AMSRD ARL CI OK TP (BLDG 4600)

NO. OF
COPIES ORGANIZATION

ABERDEEN PROVING GROUND

8 DIR USARL
 AMSRD ARL CI CT
 P JONES
 R KASTE
 A NEIDERER (5 CPS)
 M THOMAS