A Language to Model Animation of Behaviour-Embedded Graphical Components

Presented by
Shahid Alam
salam3@connect.carleton.ca

To
Professor WonSook Lee
For course ELG 7187 (Computer Animation)
@ University of Ottawa
Outline

• Introduction
• Overview of the language
• Related developments
• Modelling in ‘V’
• Case Study
• Implementation
• Comparison with other languages
• Conclusion
Introduction

• Propose a scheme for 3D animation
• Language 'V'
• Components as behaviour scripts (example)
• Scene rendering
• Nested components
• Examples
• Fan and a Walking man
Overview of 'V'

- Procedural language
- Composite structures
- Mechanisms
- Degree of freedom
- Fixed behaviour
- Parametrized behaviour
Related developments

- **ASAS** The Actor / Scriptor Animation System developed by Reynolds 1982
- **Mira Animation System** Reported by Thalmann et al 1983
- **FRAN By Elliot** 1999 [1]
- **VRML / X3D** Virtual reality modeling language 2006 [4]
- **Maya Commercial Scripting language** 2006 [2]
- **Adobe’s Atmosphere Commercial** by Adobe 2006 [3]
Modeling in 'V'

Pedestal Fan
- Motor and assembly
- Three blades
- Mounted on a shaft
- Stand
- A wide base
COMPOSITE motor {
    OBJECT motorShape;
    TRANSFORMATION orientMotor;
    COLOR lightYellow;

    motorShape = SURFACE_OF_REV ((0,0), (4,0), (13,1), (15,5),
                                  (15,15), 14,30), (13,35), (12,40), (9,45), (5,50) (0,50));

    orientMotor = RX (90) * TZ(-18);
    lightYellow = RGB (1.0, 1.0, 0.2);
    motor = orientMotor : lightYellow : motroShape;
};
COMPOSITE stand {
    OBJECT base, supportRod;
    TRANSFORMATION placeStand;
    COLOR lightBlue;

    base = SURFACE_OF_REV ((0,0), (40,0), (40,5), (35,5), (32.5,15),
                             (25,18) (12.5,24), 7.5,27), (7,30), (0,30));

    supportRod = CYL (5, 150);
    placeStand = TZ (-150);
    lightBlue = RGB (0.4, 0.4, 1.0);
    stand = placeStand : lightBlue : [supportRod + base];
};
Modeling in 'V' (composite struct blade assembly)

COMPOSITE bladeAssembly {
    OBJECT blade, shaft;
    TRANSFORMATION orientShaft, orientBade, orientAssembly;
    COLOR lightGreen, darkRed;

    orientShaft = RY (90) * TZ (-5.5);
    orientBlade = RY (-80);
    orientAssembly = RZ (90);
    lightGreen = RGB (0.2, 1.0, 0.2);
    shaft = orientShaft : darkRed : CYL (4,6);

    blade = orientBlade : lightGreen : PRISM (1, (2,2), (14,30), (15,40),
    (9,50), (0,49), (-10,45), (-15,35), (-2,2));

    // 3 copies of blade rotated 120 degrees
    bladeAssembly = orientAssembly : [shaft + REPEAT (blade, RX (120),
    3)];
};
MECHANISM fan (IN theta, IN alpha) {
    TRANSFORMATION placeBladeAssembly, spin, turn;

    placeBladeAssembly = TY (24);
    spin = RY (theta);
    turn = RZ (alpha);
    // Fan contains a stand and a motor with
    // a blade assembly which rotates alpha degrees
    fan = stand + turn : [motor + placeBladeAssembly : spin : bladeAssembly];
} fan1 (0, -60);

// Fan mechanism is instantiated with initial values
// and it animates by changing the values of alpha and theta
SCENE fan1;
Modeling in 'V' (mechanism embedded behavior)

MECHANISM fan (IN speed) {
    // Can be changed locally
    VARIABLE theta = 0, alpha = 0; thetaStep = 20, alphaStep = 0.5, dir = 1;
    TRANSFORMATION placeBladeAssembly, spin, turn;

    BEHAVIOR-SCRIPT {
        theta = theta + speed * thetaStep;
        if (theta > 360) theta = theta - 360;
        if (alpha + alphaStep > 60) dir = -1;
        elseif (alpha-alphaStep < -60) dir = 1;
        alpha = alpha + dir * alphaStep;
    }

    placeBladeAssembly = TY (24);
    spin = RY (theta);
    turn = RZ (alpha);
    fan = stand + turn : [motor + placeBladeAssembly : spin : bladeAssembly];
};
Modeling in 'V' (parameters type)

- Three types of parameters
  - **IN** A higher level mechanism can change this parameter to affect the behavior of the lower level mechanism
  - **OUT** A higher level mechanism can receive notification from lower level mechanism of any special event by using these arguments
  - **INOUT** This provides both the above functionality's in one
Modeling in 'V' (Riders)

- Moving from one frame of reference to another
- Example of conveyor belt and robot
- 4 degrees of freedom
Modeling in 'V' (Riders usage)

RIDER rGrip;

rGrip->pellet;
rGrip->null;

// We can also remove
// the object from one
// RIDER and place it
// on another RIDER
toRider->fromRider;
fromRider->null;
Case Study (group of mobile robots)

- **FreeRange[8]** Value of eight equal angles
- Goal coordinates goalX and goalY
- Robot position mrlX and mrlY
Case Study (group of mobile robots)

**MECHANISM mRob** (IN freeRange[8], INOUT {mrlX, mrlY}, IN {goalX, goalY}) {

BEHAVIOR-SCRIPT {

<< Determine next step of motion of this mobile robots using the knowledge of freeRange[] and goal and update mrlX and mrlY accordingy >>
}

<< Geometric description of mobile robot >>
};
Case Study (group of mobile robots)

**MECHANISM environ** {

**VARIABLE**

\[ n_{Rob} = 5, \quad n_{Obs} = 12, \]

\[ \text{obsRad}[n_{Obs}] = (...), \quad \text{obslX}[n_{Obs}] = (...), \quad \text{obslY} = (...), \]

\[ \text{freeRange}[n_{Rob}][8], \]

\[ \text{mrlX}[n_{Rob}] = (...), \quad \text{mrlY}[n_{Rob}] = (...), \]

\[ \text{goalX}[n_{Rob}] = (...), \quad \text{goalY}[n_{Rob}] = (...); \]

\[ \text{mRob} \quad \text{mRobot}[i] \quad (\text{freeRange}[i][8], \quad \text{mrlX}[i], \quad \text{mrlY}[i], \quad \text{goalX}[i], \quad \text{goalY}[i]), \quad i = 1 \text{ to } n_{Rob}; \]

**BEHAVIOR-SCRIPT** {

<< Compute freeRange[8] for each robot using obsRad[nObs], obslX[nObs], obslY[nObs], mrlX[nRob] and mrlY[nRob]; Redefine some or all elements of goalX[nRob] and goalY[nRob] if so desired >>

}  

<< Geometric description of environ in terms of mRobot[i] located at mrlX[i], mrlY[i], i = 1 to nRob and obstacles at obs[j], j = 1 to nObs >>

};
Implementation of 'V'

- **Language** used C using OpenGL library
- **Operating system** IRIX and Windows 9x
- **Hardware** Silicon Graphics Workstation and PC
- Scripts are supported by an interpreter
- **Components** communicate with each other using parameters of type IN, OUT and INOUT
- For making components **reusable** global variables are not used
Implementation of 'V' (flow of control)

- **Animation runs as a sequential program (OpenGL)**
- For rendering it processes from lower to higher
- Behavior scripts are processed in reverse order
- **Events are processed synchronously by all the scripts with the exception of SELECTION-SCRIPT**
## Implementation of 'V' (component comparison)

<table>
<thead>
<tr>
<th>Graphical Components 'V'</th>
<th>Software components [10]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Can't run on different machines</td>
<td>Can run on different machines on the net</td>
</tr>
<tr>
<td>Run as one sequential program, only the scripts (which are non-graphical in nature) can run as separate processes</td>
<td>Run as separate processes</td>
</tr>
<tr>
<td>Polling for checking for events</td>
<td>Event based programming</td>
</tr>
</tbody>
</table>
## Comparison with other languages

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type of animation</strong></td>
<td>Procedural animation</td>
<td>Web based modeling and animation, more in the form of presentations</td>
<td>All animations</td>
<td>Procedural and interactive animation</td>
</tr>
<tr>
<td><strong>Script Interpreter</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Data structure for animation</strong></td>
<td>Organized as hierarchy of components</td>
<td>Component based, in the form of a DAG (direct acyclic graph)</td>
<td>NA</td>
<td>Hierarchical modeling in the form of graphics state (similar to OpenGL)</td>
</tr>
<tr>
<td><strong>Universality in context of modeling &amp; animation</strong></td>
<td>No prove for its use as a general purpose language for all animation</td>
<td>General purpose language</td>
<td>General purpose language but used only with Maya</td>
<td>General purpose language</td>
</tr>
<tr>
<td><strong>Features supported</strong></td>
<td>Geometrical modeling, transformation</td>
<td>Geometrical modeling, lighting, texture mapping, transformation, NURBS</td>
<td>Lot of features</td>
<td>Geometrical modeling, transformation, camera</td>
</tr>
<tr>
<td><strong>Tool support</strong></td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Yes Text based authoring tools</td>
<td>Used with Maya</td>
<td>Built in tools for interactions</td>
<td></td>
</tr>
<tr>
<td><strong>Real-time</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td><strong>Universal rendering</strong></td>
<td>No Runs as one sequential program as one process on one machine</td>
<td>Yes Scene can be rendered out of assets located on network</td>
<td>Yes Also includes the ability to render on an unlimited number of networked machines</td>
<td>NA</td>
</tr>
<tr>
<td><strong>Support for plugins</strong></td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Conclusion (Pros)

• It requires little input from the user to generate life like motion
• A script interpreter is written to interpret the script
• Animation is little different and noteworthy
• May find use in modelling mechatronic devices (e.g. robots)
• Entire animation can be organised in terms of a hierarchy of independently graphical components
• Serve as the basis for the development of component based animation system
Conclusion (Cons)

• GUI based interactive animation has not yet been explored or implemented
• Only works with geometrical shapes
• Only appropriate for machines and mechanisms
• May not be appropriate for modelling all kinds of animations
• Lack of support for importing external images
References


