StruMML and RDS

Towards a dynamic and visual WWW

C.D. Jones

StruMML Is designed to be a dynamic map representation language. It allows for a three-dimensional environment largely based off of the architecture of the world wide web. Where the WWW consists of individual pages that are linked together through hyperlinks, StruMML consists of individual maps with hexagonal boundaries that are linked together through hexagonal hyperlinks. Each map is made up of standard tags that provide information about the models that are in it, be it terrain, npcs, or static objects. Other tags such as DIV tags and SCRIPT tags which allow for dynamic map manipulation. The entire world wide web can be a network of linked honeycomb maps.

Each StruMML resides on its respective server. For each model within the StruMML map a special series of files exist. These files are called RDS files, or Relocatable Data Sheets. They provide XML structured data about the objects in the StruMML map they are associated with. For example, an object like a boulder may have an RDS file that describes its physical properties. For more complicated map objects, like an enemy NPC , there would be an RDS for its physics as well as its AI.

RDS files are called relocatable because they can be transferred from server to server, where a StruMML file is static and must remain on its home server . RDS files can contain all the info needed by the server, such as physics, AI, animation, etc. or those tags can be separated onto different files on the server.

If we were to examine our own minds and the world around us, we would see that our brains are similar to web browsers. Every creature has a different perception of the world around them, just as each web browser can portray a different web page differently. In this case, our brains browse through a 3D environment. The three dimensional environment is represented by StruMML pages which our StruMML browser is able to represent. The laws of physics on the other hand, is integrated into our universe and the same for every browser that view it. This is why the laws of physics are handled on StruMML server-side.

The true power of the Virtual World Wide Web and StruMML is that it is distributed. Because the actual VWWW will be distributed across hundreds, thousands, maybe even millions of servers, each individual server only must handle a small part of the data. This means that the server will have enough processing power to dynamically alter the StruMML map according to the respective instructions of the browser and the server as the RDS changes. The majority of the processing power of the browser will handle rendering (which includes caching adjacent maps) and transmitting user interactions with the environment to the server.

Animation

Animation is handled by the server. The server modifies the model attribute in the NPC tag to point to the current model being used for animation.

**Life Cycle**

1. Generate a request by browser
2. Server receives request and opens comm channel
3. Server continues RDS maintenance cycle
4. Server sends home StruMML to browser and appropriate snapshots
5. Browser displays StruMML map along with all models
6. Browser awaits input.
7. Server maintains RDS maintenance cycle

**RDS Maintenance Cycle**

If collision, update RDS

Update

StruMML

Scan all objects in StruMML

For all objects

If in motion/animation update motion/anim

Update Snapshot

**Server Operations**

Browser Communication Thread

StruMML maintenance Thread

Server Process

StruMML maintenance Thread

Browser Communication Thread

RDS Maintenance Thread

RDS Maintenance Thread

RDS Maintenance Thread

**Example StruMML Code**

//a link to a hexagram map

<A HEXREF=<http://www.server.com/maps/gorgemap.smml>” cord=”0,0,1” />

//information concerning a character. Sort of a blueprint

//alterable by an RDS

<NPC id=”00abc” name=”Machine Gunner Bob”

physics=”<http://www.server.com/physics/gunerbob.rds>”

//automatically queries for current animation snapshot.

model=”http://www.server.com/models/gunnerbob.snapshot”

//ai information

ai=”http://www.server.com/ai/gunnerbob.rds”/>

</NPC>

//a link to a static model. RDS represents any change to the base terrain map

<STATIC id=”00c” name=”Tree” physics=”<http://www.server/terrain1.rds>”

model=”<http://www.server.com/terrain/gorgemap.terrain>”>

**Example RDS Code**

//rds is tied to strumml code by ids

//rds files contain multiple physics tags, animation tags, etc.

//or only one tag

//Objects are generic tags for hierarchies

<Object id=”00abc” name=”Machine Gunner Bob” type=”complete”>

<Physics>

<mass>5 kg </mass>  
 <length> 3m </length>

<height> 10m </height>

<width> 10m </width>

</Physics>

<Animation id=”001” name=”Running” frames=”10”>

<skeleton>http://www.server.com/skeletons/gunnerbobrunning.skel/>

//the tag below shows that a new object is produced from the animation

<produces>http://www.server.com/physics/dustcloud.rds/>

</Animation>

<Animation id=”002” name=”Shot” frames=”4”>

<skeleton>http://www.server.com/skeletons/gunnerbobshot.skel/>

//in this case the skeleton produces a new object, blood puddle, that is defined later.

<produces type=”object”>#00bc</produces> //works like a link jump on a //page

</Animation>

</Object>

<Object id=”00bcb” name=”Blood Puddle” type=”complete”

<Physics>

<mass> 1 g </mass>

<length> 10 m </length>  
 <depth> 2 cm </depth>

<state> liquid </state>

<width> 10m </width>

</Physics>

</Object>

**Object Interaction Detection**

**(Requirements for a StruMML Browser)**

Every unforeseeable circumstance in a simulation or video game begins with one thing, object interaction. Sometimes known as collision detection, object interaction can break or make a simulation.

Simple collision detection is easy. Just wrap a rectangular prism around two objects and see if they intersect. Easy? However, we need a game and physis engine that will know to adapt to new situations. We need an intelligent game engine and an intelligent physics engine.

The following are two simple rules that we would like our game engine to do without coding and their requirements.

Game Engine allows all rocks to be thrown.

-How do I know if it’s a rock? Requires a dynamic game engine

-How do I know if I can pick it up? Requires a dynamic game engine

-What does it look like when I throw it? Requires a dynamic game and skeletal engine

-Where will it land? Requires an intelligent physics engine

Game Engine a plane to explode and put a hole in the ground.

-Where does the shrapnel go? Requires an intelligent physics engine

-Where does the shrapnel come from? Requires a dynamic model generation engine

-How is the terrain deformed? Requires a dynamic terrain deformation engine

-What do the particles from the crash look like? Requires a dynamic particle engine

**More RDS Code**

<Object id=”00abc” name=”Generic Cube” type=”complete”>

<Physics modifier=”stationary”>

<weight> 5 kg </weight>

<length> 5 cm </length>

<width> 5 cm </width>

<height> 5 cm </height>

<density state=”solid” metal=”true” > 51kg cubed</density>

</Physics>

//position of center of gravity

<Coordinates>

<xpos> 0 </xpos>

<ypos> 0 </ypos>

<zpos> 0 </zpos>

</Coordinates>

<Physics modifier=”movement”>

<velocity> 0 </velocity>

<acceleration> 0 </acceleration>

</Physics>

<AI>

<PathFinding> False </PathFinding>

<Trackable>True </Trackable>

</AI>

Density Tag in the RDS determines whether a model is hollow, liquid, sand, etc. and how the engine handles it.

Based off of RDS info and calculations from input, the engine will calculate and generate in real time deformations. These models will be updated as snapshots of the original models.

Requires a dynamic, model generation engine. This engine creates particles and objects such as shrapnel, gunfire, etc. It also creates places generated on the fly.

For example. If a person is wounded and loses a leg, the DMG engine (Damage Engine) will show creature lost a limb and add it to the scene using its AI and knowledge of its previous limb severed.

DMG Engine and DEFORM engine work using an AI that records the scenarios in previous model physics interactions and redirected them when there is not enough info (model data) present.