Introduction

Genesis is a Java-like language that allows for the easy creation of 2D games.

This language utilizes a collision operator to abstract away the logic behind collision detection so that the programmer can focus on the game mechanics.

It also include a complex data type “cluster” that serves as the fundamental building block for game design.

Data Types:

**Primitive data types (Passed by value)**

- int
- float
- char
- boolean

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>integer</td>
</tr>
<tr>
<td></td>
<td>Example: 100</td>
</tr>
<tr>
<td>float</td>
<td>float</td>
</tr>
<tr>
<td></td>
<td>Example: 2.3</td>
</tr>
<tr>
<td>char</td>
<td>character</td>
</tr>
<tr>
<td></td>
<td>Example: ‘a’</td>
</tr>
<tr>
<td>boolean</td>
<td>Boolean value (true or false)</td>
</tr>
<tr>
<td></td>
<td>example: true</td>
</tr>
</tbody>
</table>
Complex data types (Passed by reference)

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pixel</td>
<td>X value, Y value, color</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td>color</td>
<td>R value, G value, B value</td>
</tr>
<tr>
<td>cluster</td>
<td>1. Collection of points and/or previously defined clusters which form a</td>
</tr>
<tr>
<td></td>
<td>geometric shape and it's accompanying color.</td>
</tr>
<tr>
<td></td>
<td>2. Includes an anchor point to reference the shape.</td>
</tr>
<tr>
<td>array</td>
<td>A container that holds data, dynamically sized</td>
</tr>
</tbody>
</table>

Language Components:

Control Flow

- if
- elif
- else
- while
- for

Game Specific

- START - creates and displays a gameboard with provided dimensions, pixel size, and background color.

Indexing

- Indexing starts at 0
- Subarrays are indexed by brackets [] with a : separating first and second indices similar to Python. The first index is inclusive, and the second index is exclusive. For example array[1:5] includes the element at index 1 until but not including the element at index 5.

Function

- func returnType funcName(args) { }

Comments

- Comment Symbol: //
Logical Operators
- AND
- OR
- NOT

Comparison Operators
- Equals: ==
- Not equals: !=
- Greater than or equals: >=
- Less than or equals: <=
- Greater than: >
- Less than: <

Math Operators
- Multiply: *
- Divide: /
- Add: +
- Subtract: -
- Modulo: %

Collision-to-event Operator
- clust1 <!> clust2 returns whether clusters clust1 and clust2 are currently touching (in adjacent pixels) or overlapped

User Input
- keyPressed(KEYNAME) returns whether KEYNAME is pressed
- keyReleased(KEYNAME) returns whether KEYNAME was released since the last frame

Time & Clock Speed
- Timing is controlled by providing deltaTime, the time that has elapsed since the last frame was rendered, through the onUpdate function.
- The engine will render frames as quickly as possible (Clock speed is not defined by the programmer or the language)

**Code Example**

- In the background grab the screen resolution/dimensions and scale appropriately
  - Initialize dimensions n * m
  - Initialize square pixel size x * x
  - Initialize background color <r,g,b> (Prim)
  - Define timer: paired with seconds on a clock
  - User inputs (keys &/or mouse)
  - Define shapes/color
// Make new board
START 1280, 960, 1, <255, 255, 255>

// Make new object cluster
cluster lavaWall = new cluster{
    x=1080;
    y=480;
    width=100;
    height=800;
    color = <255, 0, 0>;
}

float timePressed = 0;
cluster player = new cluster{

    //These properties are of type int
    x=200;
y=480;
    width=100;
    height=100;
    color = <0, 0, 255>;

    // Automatically gets called by engine
    onUpdate(float deltaTime) {
        if(keyPressed("right")) {
            timePressed += deltaTime;
            if(timePressed > .1){
                //Move if the key has been pressed for .1 sec
                this.x += 1;
                timePressed = 0;
            }
        } else {
            timePressed = 0;
        }

        //Check for collision
        if(this<!>lavaWall){
            this.delete();
        }
    }
}

//Global update function
onUpdate(float deltaTime){
    if(keyPressed("esc")){
        wall.delete();
        player.delete();
    }
}