Genesis

Created by: Jason Zhao, Leo Stilwell, Michael Wang, Saahil Jain, Sam Cohen

A language for implementing interactive 2D-games.

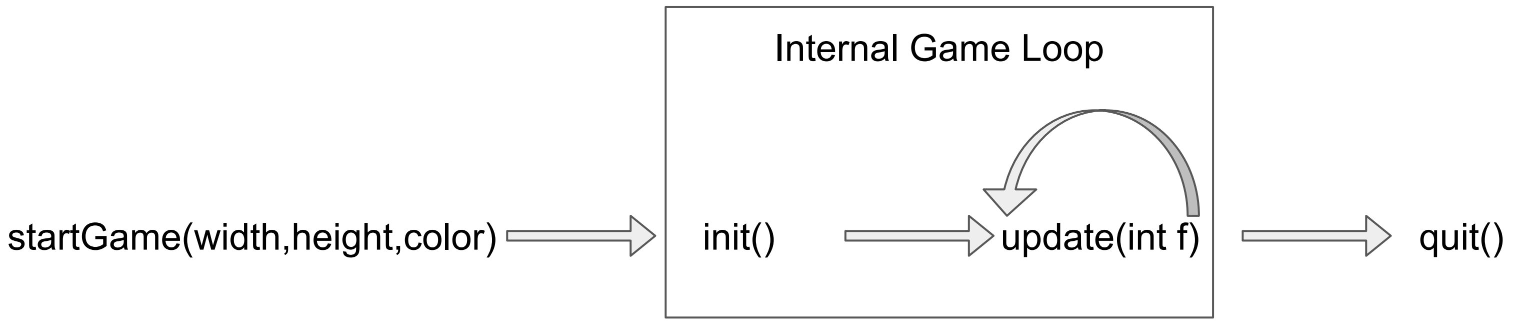
Language Features

Language Features

- Genesis is designed to be intuitive and expressive for game developers, without all the frills.
- Genesis abstracts away the game engine, allowing developers to simply define the objects and their associated interactions. No need to touch C or a graphics library!
- Genesis provides a simple way to do everything from defining colors and clusters to initializing screens.
- Genesis runs on top of an update function that handles game behavior, enabling the creation of dynamic, engaging games.
- Genesis provides a robust array built with game design in mind.

Making a Game

Game Operation



void init()

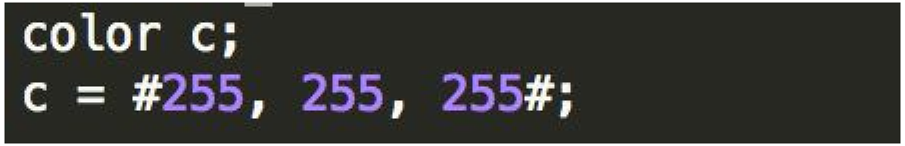
Called immediately after the game window has been created, before any frames have been rendered.

void update(int frameNumber)

Called every time a frame is rendered, and takes in an integer value that represents the total number of frames that have been rendered so far.

Colors

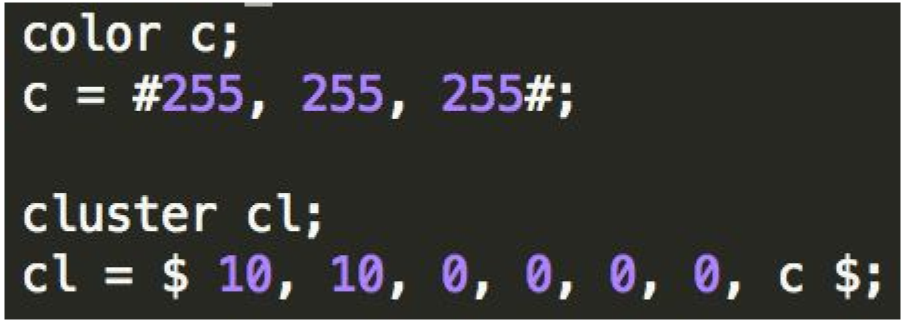
A primitive type that consists of three integers that represent r, g, and b values. The following lines of code represent the color white.



```
color c;
c = #255, 255, 255#;
```

Clusters

Objects that represent rectangular clusters of pixels. They must be initialized with initial width, height, x, y, dx, dy, and color values:

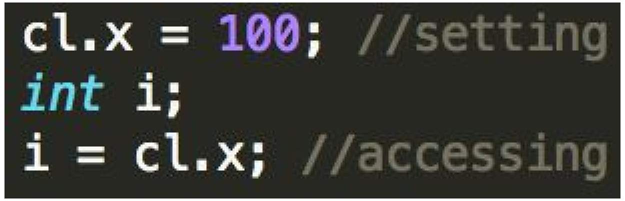


```
color c;
c = #255, 255, 255#;

cluster cl;
cl = $ 10, 10, 0, 0, 0, 0, c $;
```

Cluster Properties

Properties of colors can be set and accessed using the ‘.’ operator, like so:



```
cl.x = 100; //setting
int i;
i = cl.x; //accessing
```

Property name	Property Type	Description
width	int	Width, in pixels
height	int	Height, in pixels
x	int	X position, in pixels
y	int	Y position, in pixels
dx	int	X velocity, in pixels per frame
dy	int	Y velocity, in pixels per frame
color	color	The color of the cluster
draw	bool	Whether the cluster should be displayed

Key Input

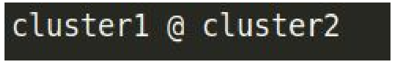
Users can monitor whether a key has been:

- Pressed for the first time - keyDown()
- Held down - keyHeld()
- Released - keyUp()

Each function takes in the name of the key and whether the given state is currently true.

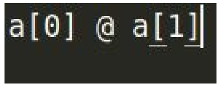
Collision Detection

- Simple Syntax



```
cluster1 @ cluster2
```

- Easy to check even in an array



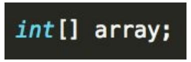
```
a[0] @ a[1]
```

- returns a boolean value - true if the clusters collide, false if they don’t

Arrays

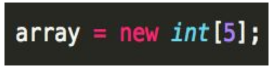
Genesis provides an array type that is crucial to implementing various game features.

- Array declaration syntax:



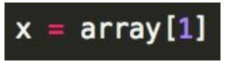
```
int[] array;
```

- Array initialization using the *new* keyword:



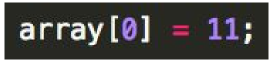
```
array = new int[5];
```

- Array Access:



```
x = array[1]
```

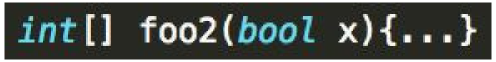
- Array Assign:



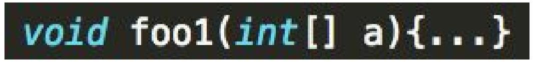
```
array[0] = 11;
```

Arrays

- We noticed that many other projects implemented arrays whose type was bound to their size. Instead we implemented a size-agnostic array that uses pointers-- allowing arrays to be passed back and forth between functions with ease.



```
int[] foo2(bool x){...}
```



```
void foo1(int[] a){...}
```

- Arrays can hold all data types, but are not recursive.

Miscellaneous Functions

int random(int max)
Returns a random integer in the range [0, max)

setFPS(int fps)
Sets the rate at which frames are rendered and the update() function is called. The default fps is set at 60.

Test Suite & Building

- Cross-platform development cycle
- Split tests into regression and new tests
- LetThereBe.sh

Demo