Final Report

Members:

- Fernando Luo **(fbl2108)**
- Papoj “Hua” Thamjaroenporn **(pt2277)**
- Lucy He **(lh2574)**
- Kevin Lin **(kl2495)**

1. Introduction

1.1 Language Overview

RetroCraft is a programming language that aims to provide users with the tools to easily and creatively design a computer game. Our language focuses specifically on side-scrolling, obstacle-aversion style games. Games produced would be similar to Helicopter: a simple platform game in which the player has to keep a helicopter flying through a generated scene as far as possible without being hit by obstacles. Our language supports basic and more advanced functionalities including arithmetic operations, control flow, user-defined functions, recursion, and arrays of primitive types and scene objects. Combining built-in objects and functions that assist the coder's creative process with imagination and intuitive code, our language is a powerful tool that casual gamers can easily use to generate their own game with impressive results.

1.2 Background

Since the creation of platform games in the 1980s, video gamers have witnessed the growth and evolution of 2D platformers. The genre persists today with various legacies of games such as Super Mario Bros and Donkey Kong. However, gamers and hobbyists rarely have the chance to design their own. We have implemented a language that provides users with the building blocks to conveniently and creatively design their own game level, specifically for a game of a similar kind to Helicopter. RetroCraft defines an intuitive syntax that will allow the programmer to express the boundaries of a level, scene generation mechanics, and player characteristics. The language also provides powerful built-in functions that will execute game mechanisms without any specification from the user. These features include: collisions detection of generalized polygons, infinite loops that update the scene, the image generation mechanism, the score of the player, and the input events that detect keyboard input and respond accordingly automatically.

2. Language Tutorial

2.1 File Extension

Our language executes source code with ".rc" extension.

2.2 Compiling and Running Test Cases

Our language comes with a Makefile that can be used to easily compile our language compiler. To run the source code, execute:

```
./retrocraft [options] < [.rc files]
```

Options:

```
-b    Generate the byte code
-c    Compile the source code (default)
```

2.3 Generating Test Cases Reference

We provide a shell script `testall.sh` which can be executed to either: generating test case references, or running the testing source codes in the test suite against the references. The command is the following:

```
./testall.sh [options] [.rc files]
```

Options:

```
-k    Keep intermediate files
-r    Generate test references instead of running code
against them
-h    Print this help
```

If the file is not specifies, the script will run the code through all source codes that live within the main directory. Please note that to be able to test the codes or generate the references, one must `make` first

2.4 A Simple Program: Greatest Common Divisor

The following program evaluates the greatest common divisor of a given set of three integer pairs. Through this sample, we demonstrate the concept of user-defined functions, function calls, and flow control (an `if` statement and a `while` loop).

```
function $gcd : (int $a, int $b)
{
     while ($a != $b)
     {
          if ($a > $b)
               $a -: $b;
          else
               $b -: $a;
     }
     return $a;
```

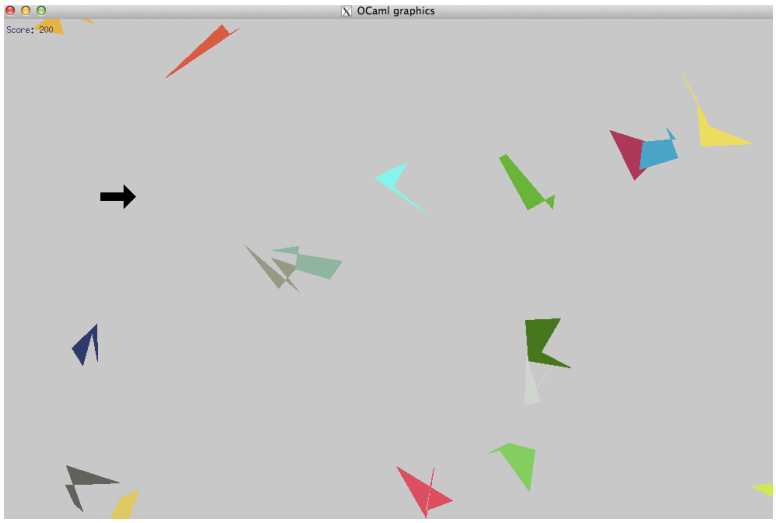
```
    }

    function $main : ()
    {
        $printstring("Should print 2, 3, and 11");
        $printint( $gcd(2,14) );
        $printint( $gcd(3,15) );
        $printint( $gcd(99,121) );
    }
```

2.5 A Simple Helicopter Game

```
/* Create global Map object */
Array int $vertices;
Array Brick $b;

function $generate : () {
    int $i; int $j;
    Brick $b1;

    for ($i : 0; $i < 20; $i +: 1) {
        for ($j : 0; $j < 5; $j +: 1) {
            $vertices[$j*2] : $GenerateRandomInt(100);
            $vertices[$j*2+1] : $GenerateRandomInt(100);
        }
        $b1 : new Brick($generateRandomColor(),
$generateRandomColor(), $generateRandomColor(), $vertices,
$GenerateRandomInt(1000), $GenerateRandomInt(700));
        $Push($b, $b1);
    }
    $printint($ArrayCount($vertices));
    return $b;
}

function $generateRandomColor : () {
    return $GenerateRandomInt(255);
}


function $getPolygonVerts : (int $sx, int $sy, int $size)
{
    Array int $verts;
```

```
     $verts : new Array int;

     $verts[0] : $sx                              ; $verts[1] :
$sy + ($size / 3);
     $verts[2] : $sx  + $size                ; $verts[3]: $sy +
($size / 3);
     $verts[4] : $sx + $size                 ; $verts[5] : $sy;
     $verts[6] : $sx + (3 * $size / 2)  ; $verts[7] : $sy +
($size / 2);
     $verts[8] : $sx + $size                 ; $verts[9] : $sy +
$size;
     $verts[12] : $sx                             ; $verts[13] :
$sy + (2* $size / 3);
     $verts[10] : $sx + $size           ; $verts[11]: $sy +
(2*$size /3);

     return $verts;
}

function $main : () {
     Map $myMap;
     Player $p;
     Array int $pv;
     int $i; int $size; int $startX1; int $startY1;
     int $startX2; int $startY2;

     $size : 30;
     $startX1 : 100; $startY1 : 200;
     $startX2 : 500; $startY2 : 0;

     for($i : 0; $i < 5; $i +: 1) {
          $Push($pv, $GenerateRandomInt(60));
          $Push($pv, $GenerateRandomInt(60));
     }

    $p : new Player(0, 0, 0,$getPolygonVerts($startX1 + ($size),
$startY1 + ($size), $size),$GenerateRandomInt(700));
     $myMap : new Map(1000,700,$generate);
```

```
	$Run($myMap, $p);
}
```



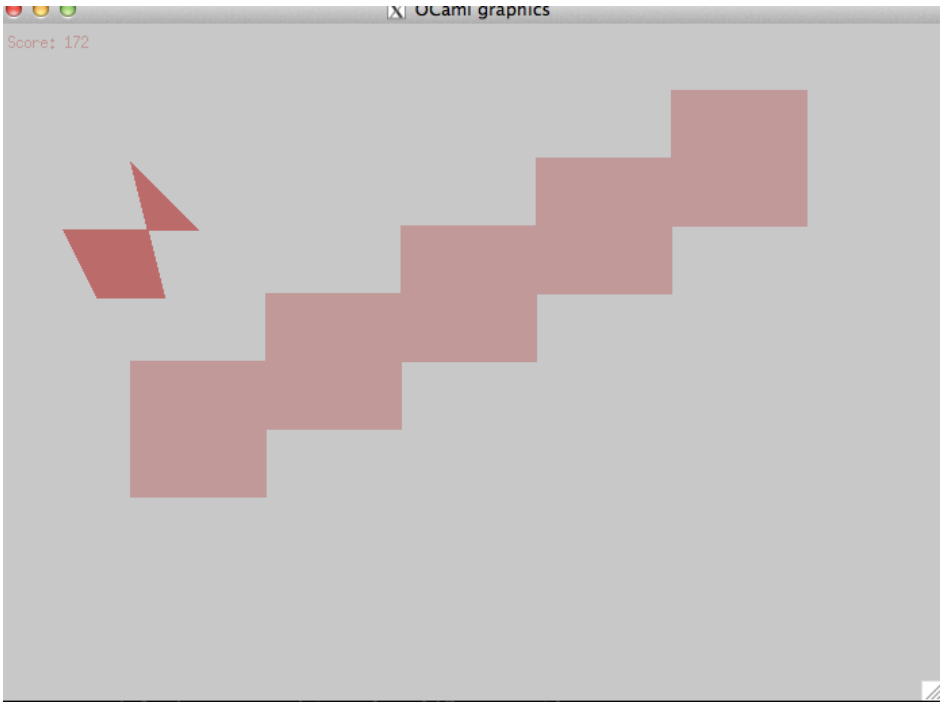
User is controlling the spaceship (arrow) using the space key to ascend. As objects hurl towards the user, the user need to dodge the onslaught of generated obstacles.

2.6 A More Complex Sample: Generating the Obstacles

```
/* Create basic game with stair like brick obstacles */

function $translateRect : (Array int $arr, int $x, int $y) {
	Array int $temp;
	int $i;

	$temp : new Array int;

	for ($i:0; $i<8; $i+:2){
		$temp[$i] : $arr[$i] + $x;
		$temp[$i+1] : $arr[$i+1] + $y;
	}

	return $temp;
}

function $generate : Array Brick () {
	Array Brick $br;
	Array int $b;
	Array int $c;
```

```
	int $j;

	$b : new Array int;
	$br: new Array Brick;

	$c : new Array int;
	$c[0] : 250;
	$c[1] : 100;
	$c[2] : 350;
	$c[3] : 100;
	$c[4] : 350;
	$c[5] : 200;
	$c[6] : 250;
	$c[7] : 200  ;

	for ($j:0; $j<5; $j+:1){
		$b : $translateRect($c, $j*100+100, $j*50);
		$br[$j] : new Brick (200,150,150,$b,50,50);
	}

	return $br;
}

function $main : () {
	Map $myMap;
	Player $p;
	Array int $v;

	$v : new Array int;

	$v[0] : 75;
	$v[1] : 100;
	$v[2] : 50;
	$v[3] : 150;
	$v[4] : 150;
	$v[5] : 150;
	$v[6] : 100;
	$v[7] : 200;
	$v[8] : 125;
	$v[9] : 100;
```

```
    $p : new Player(200,100,100,$v,50);
    $myMap : new Map(700,500,$generate);

    $Run($myMap, $p);
}
```



3. Language Reference Manual

3.1 Lexical Convention

3.1.1 Comments

Double forward slashes `//` indicate the beginning of a single line comment. Multiple line comments will begin with `/*`and end with `*/`.

3.1.2 Tokens

The types of tokens in our language are: keywords, identifiers, constants, string literals, operators and separators.

3.1.2.1 Keywords

RetroCraft has a list of reserved words with fixed purposes.
Variable type declaration: `int, string, function, void`
Control flow: `if, else, while, for, return, true (1), false (0)`

Data object: `Array, Map, Player, Brick`

3.1.2.2 Identifiers

Identifiers begin with a dollar sign (`$`) followed by a sequence of upper and/or lowercase characters, digits and underscores, starting with a non-numerical character. The keywords in 2.2.1 are not valid identifiers. Upper and lower case characters are unique, making identifiers case-sensitive.

3.1.2.3 Separators

`\t`	tab
`\n`	new line feed
`\r`	return
`<space>`	space

3.1.2.4 Punctuators

`;`	end of line
`,`	separates arguments, object attributes
`{ }`	code block
`" ... "`	double quotes for `string`
`()`	function calls or arithmetic operations
`[ ]`	array random access
`.`	referencing object's attributes and functions

3.1.3 Operators

3.1.3.1 Arithmetic

Our arithmetic operators will be the standard operators present in most languages. The symbols and associated operations are as follows:

:	Assignment
+, -	Addition and Subtraction
+:, -:, *:, /:	Shorthanded Add, Subtract, Multiply, and Divide
*, /	Multiplication and Division
%	Modular

Arithmetic expressions will be made using infix notation, i.e. `operand1 operator operand2`. The standard order of operations specified by arithmetic will be honored, i.e. PEMDAS. Arithmetic operates on type `int`.

3.1.3.2 Comparison

=	Equal
!=	Not equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal

These operators compare variables and/or constants with each other and return an integer constant (`1` for true, and `0` for false). Incompatible types will result in a syntax error.

3.1.3.3 Logical Operators

&&	AND
||	OR
!	NOT

Logical operators can be used with expressions which evaluate to either `1` or `0`. The order of precedence is: NOT, then AND and OR. It is recommended that a parenthesis is used when an expression involves multiple logical operators, e.g., `($x = 3) || (($x = 4) && ($y = 1))` instead of `($x = 3) || ($x = 4) && ($y = 1)`

3.1.3.4 Member Operators

Member operators on objects will use a single dot (.) notation. For example, to access the `$height` property of a Map object `$gameMap`, the notation `$gameMap.$height` should be used.

Member operators on our zero based arrays will use a square bracket notation. For example, to access the 2nd index of an array `$sampleArray`, the notation `$sampleArray[1]` should be used.

3.2 Statements

3.2.1 if, else if, else

`if`, `else if` and `else` statements are used to control when their contained blocks of code will be executed. For example:

```
if (<logical expression>) {
     // code executed if above expression evaluated to true
} else if (logical expression) {
     // code executed if first logical expression was false
     // and the second was true
} else {
     // code executed if both logical expressions were
false
}
```

3.2.2 for

`for` statements are used to control the number of times a block of code is executed. The for statement has three components:

```
     for (<variable initialization> ; <logical expression> ;
<variable increment/decrement>) {
          // code to execute
     }
```

The code will continue to be executed as long as the logical expression is true. The variable initialization and increment/decrement give a compact way to control the number of times the code is executed. Code block following the for statement must be wrapped in brackets.

For example, the following would iterate through the code 5 times:

```
int $i;
for ($i : 0; $i < 5; $i +: 1) {
    // code to execute
}
```

3.2.3 while

A `while` loop evaluates the bracketed statements if the given logical expression remains true.

```
while (<logical expression>) {
    // code to execute
}
```

3.2.4 return

Functions terminate when they reach a return statement.

3.3 Declarations and Assignments

3.3.1 Primitives

RetroCraft supports two primitive types: `int` and `string`. We can declare a new primitive variable using the following syntax:

```
// Declaration and Assignment done separately
<primitive type> $<var_name>;
$<var_name> : <value>;
```

For example:

```
int $myInt;
$myInt : 5;
```

There is one thing we need to point out regarding the `string` type. According to how we designed the memory allocation, we have decided to allocate 40 words (1 word = 4 bytes) on the stack for a string. For this reason, the user will be able to use the string variable safely as long as the length of the string is not longer than 38 characters (the other two words are necessary for bookkeeping purposes on the stack). **Section 5** will discuss more about the architecture design.

3.3.2 Arrays

RetroCraft fully supports arrays of all types (`int, string, Brick, Player, Map`). Similarly to with primitives, an array must be declared first, and then initialized using the keywords `new Array <type>`. To access or define elements in an array, we use square brackets containing the desired element index. The syntaxes are shown below:

```
// Declaration, allocation, and assignment done separately
Array <object_type> $<name_of_array>;
$<name_of_array> : new Array int;
$<name_of_array>[0] : <some_data>;
$<name_of_array>[1] : <some_data>;
...
```

For example:

```
Array int $arrayOfInts;
$arrayOfInts : new Array int;
$arrayOfInts[0] : 4;
$arrayOfInts[1] : 1;
$arrayOfInts[2] : 2;
```

The way we can access an array element is the following:

```
$arrayOfBlocks[1]
```

The index of any array starts from zero.

There are two aspects of the array we need to point out here. First, notice how the size of the array is never needs to be specified. We would like to simulate a dynamic array in our program. However, the actual array is always allocated 100 slots (i.e. for 100 elements, regardless of type). Therefore, the user can use the array as long as the number of elements does not exceed 100. Second, to initialize the array the keywords `new Array` are used to label pieces in memory as belonging to a certain type of array. In fact, `new` can also be used to create game objects including Map, Player, and Brick. **Section 5** will explain more about this in details.

The size of elements in the array can also be accessed by the attribute `length`. For example:

```
$countArray()
```

3.3.3 Function Declaration

Function declarations begin with the keyword `function`. The header will also contain the return type and formal parameters. If there is no return type, `void` should be used instead.

```
function func_name : (<parameters>) {
    // Implementation
```

```
};
```

For example,

```
function $retMapArray : (int $total)
{
     Array Map $retArray;
     Map $m1;
     int $i;

     $retArray : new Array Map;

     for ($i : 0; $i < $total; $i +: 1){
          $m1 : new Map (768, 1024, $generateThis);
          $retArray[$i] : $m1;
     }
     return $retArray;
}
```

We will inherit the same mechanism on parameter passes from OCaml: all parameters are implicitly passed by reference.

3.4. Primitive Data Types and Basic Data Types

Our language has five unique data types and another five data types which are just array types for the first five. These five unit types are outlined in the tables below.

3.4.1 Primitive Data types

`int`	`..., -1, 0, 1, ...`
`string`	`"Hello World"`

3.4.2 Basic Data types

`Array` (See 3.3.2)	Stores a collection of data elements of the data type. Array elements are accessed with square brackets. *Attributes* `$length` — The length of the array
`Map`	The canvas for the game. It is the container for all objects including

<table>
<tr><td></td><td>Brick and Player in the game. It also contains the generator function pointer that invokes function to build all blocks.

Variable and Object Attributes
<table><tr><td>int $width</td><td>Width of the game screen</td></tr><tr><td>int $height</td><td>Height of the game screen</td></tr></table>
Function Attributes
<table><tr><td>$generateThis</td><td>Function pointer that returns an array of blocks</td></tr></table></td></tr>
<tr><td>Brick</td><td>Fundamental building blocks of the game environment. User provides parameters: (R, G, B, $verticesArray, x, y).

Variable and Object Attributes
<table><tr><td>int $colorR
int $colorG
int $colorB</td><td>User provided RGB values of the brick.</td></tr><tr><td>int $x,
int $y</td><td>Translation coordinates of object</td></tr><tr><td>Array int $verticesArray</td><td>pointer to an array of integers (vertices array)</td></tr></table>
Brick objects will be translated along the map internally to simulate movement. Its movement is independent from Player.</td></tr>
<tr><td>Player</td><td>The user controlled character, which can be controlled to move through the map. Similar to Brick, user supplies the RGB values, pointer to the vertices array, and the starting Y position

Variable and Object Attributes
<table><tr><td>int $colorR
int $colorG
int $colorB</td><td>User provided RGB values of the Player object.</td></tr></table></td></tr>
</table>

	`Array int $verticesArray`	pointer to an array of integers (vertices array)
	`int $y`	Translation coordinates of object
	`Player` and `Brick` move independently of each other. User will be able to move the `Player` up and down (Y position).	

3.5 Operations on Graphics Objects

Since RetroCraft is primarily graphics based, we require a specific set of attributes and methods in order to control the layout and flow of the game. The following sections describe them.

3.5.1 Object Construction

Object variables are declared and constructed similar to the syntax specified in the variable declaration section above (3.3.1):

```
<object type> $<var_name>;
$<var_name> : <attributes>;
```

Instead of a primitive type, the variable name is preceded by an object type, specified as a data object keyword in section 3.1.2.1. Similar to the initialization of an Array, data object types uses the keyword `new` as well.

```
$myMap : new Map(700,500,$generate);

$b1 : new Brick(100,150,200,$vertices,20,30);
```

A detailed example:

```
function $main : ()
{
      Player $p1;
      Array int $vertices;
      $vertices : new Array int;
```

```
        $vertices[0] : 400;
        $vertices[1] : 200;
        $vertices[2] : 150;
        $vertices[3] : 300;

        $p1 : new Player(0,0,255,$vertices,10);
    }
```

To access the object and its attribute after creation, one can do a simple reference:

```
        $printint($p1.$colorR);
        $printint($p1.$colorG);
        $printint($p1.$colorB);
        $printint($p1.$y);
        /* player vertices */
        $printint($p1.$vertices[0]);
        $printint($p1.$vertices[1]);
        $printint($p1.$vertices[2]);
        $printint($p1.$vertices[3]);
```

3.5.2 Display and Movement

The game map is a grid of a user-determined height and width measured in pixels. Coordinates increment up and to the right, such that the bottom left space in the map has the coordinates (0,0). Game objects, are rectangular shaped entities specified by height and width values and are placed on the game map grid at specified coordinates according to their `$x` and `$y` attributes. Upon rendering an object, the bottom left corner of the object is placed at the specified coordinate on the game map and the rest of the object spans the space above and to the right. Our language will internally move the Brick objects to the left as it detects object collision. The user will press the spacebar in order to move the player.

3.5.3 Modifying Objects

Attributes of various objects can be modified after object creation by referencing the object (`$<object name>`) and using the punctuator **`'.'`** to call attributes:

```
function $main : ()
{
    Array int $vertices;
    Brick $b1;
    $vertices : new Array int;
```

```
    $vertices[0] : 567;
    $vertices[1] : 420;

    $b1 : new Brick(100,150,200,$vertices,20,30);

    $b1.$colorR : 255;
    $b1.$colorG : 255;
    $b1.$colorB : 255;
    $b1.$vertices[0] : 121;
    $b1.$vertices[1] : 408;
    $b1.$x : 0;
    $b1.$y : 0;
}
```

3.5.4 Advanced Attributes and Functions of Object's

The object does not only provide basic attributes such as width and height of the object, but also some functionality that, after being defined by the user, can be used to control the behavior of the object and its interaction with other objects.

3.5.4.1 Dimensions

Each object's dimension attributes, $height and $width, define the rectangular area of pixels allotted to it on the grid.

3.5.4.2 Coordinate Location

Each object's coordinate attributes, $x and $y. These coordinates could be changed over the course of a game with internal keyboard events.

3.5.4.3 generateThis (Map)

The Map object has a pointer to a function that generates and returns an array of Bricks. This function will be invoked as the game progresses to draw blocks. User can program it to dynamically change the map depending on the score.

3.6. Built-in & Required Functions

3.6.1 `main`

Every game created by RetroCraft requires a main function. All games will begin execution from this function.

The `$main()` function is composed of two main sections. The first section includes the initialization of all variables. The next section follows normal program flow; provide that any necessary initializations are done first.

3.6.2 `Run (Map $mapObject, Player $playerObject)`

The `$Run` function takes a Map and a Player object and invokes the helper built-in functions: `$DrawPlayer` and `$CallGenerator`. It builds the game with necessary bookkeeping functions and displays the game onto a graphics window.

3.6.3 `LoadPlayer (Player $playerObject)`

The `$LoadPlayer` function takes a player object and paints it on the graphics window.

3.6.4 `printint` (int $i) or printint (1)

Prints an integer literal or a integer variable onto the console. Retrocraft will type check the parameter to ensure that this function prints only data type int.

3.6.5 `printstring (string $str) or printstring("hello")`

Prints a string variable or a string literal with a maximum length of 38 characters. Retrocraft will type check the parameter to ensure that this function prints only data type string.

3.6.6 `dumpstack()`

The dumpstack function allows user to display the entire stack on console. This allows for ease of debugging and for one to access and trace through the memory structure

3.6.7 `CallGenerator (Map $mapObject)`

This function will invoke the $generator function inside the given Map object and create the block of Bricks necessary for display. This function is called automatically when $Run is invoked.

3.6.8 `Push (Array <type> $in_array, <type> $object)`

The push function will push an object into the specified array. If the array is full, an exception will be thrown

3.6.9 `GetCurrentScore ()`

A built-in function that allows user to obtain the score within a lifetime of a game and put it on top of the stack.

3.6.10 GenerateRandomInt (int $i)

User can use this function to generate a random integer using another integer as a seed. Retrocraft will type check to make sure that the parameter is indeed an integer.

4. Project Plan

4.1 Process:
The most important part of this project was to plan out the roadmap of the project. The brainstorming started in the beginning of the semester. And we decided that we wanted to do something graphical. After much debate over the semantics and the conventions of our language, we started to work on Scanner. Overall, the project was a very collaborative effort. We would normally have code on a large TV monitor and conduct several group programming session during weekends.

As we progress, we decided various flaws, inconsistencies, and just overall disagreements. Whenever we discover something we would like to remove or add, we need to go back to Scanner/Parser/AST. As a result, the initial phase was laboring, and at times, with little sense of concrete direction.

Regarding programming practices, we would always be in the same room, same time; if not actively group programming on the TV. This allows everyone to communicate with ease and address issues quickly.

4.2 Overall Timeline
September 28th – Proposal
October 31st – LRM
November 18th – Scanner
December 6th – Parser and AST (with MicroC as reference)
December 15th – ByteCode
December 16th – Graphics and Compile
December 17th – Execute, Generating testsuites
December 18th – Additional Graphics, Passed all standard tests
December 19th – Optimizing Graphics and Stack operation

Roles and Responsibilities:
Although most of the project was done together as a group, we eventually needed to split tasks as the deadline looms. As a whole, each group member actively participated in creating Scanner, Parser, and AST.

Hua started to work with graphics in OCaml while Lucy and Fernando started Bytecode. Kevin started to work with Compile. After bytecode was done, Lucy and Kevin both worked on Compile and Execute. Fernando started the test cases and added to Execute. Finally, Hua

developed the graphics engine to paint, and transform shape, while Fernando simulated game conditions such as gravity and looping of gaming maps.

Testing the code was a collaborative effort with contributions from every member. Hua and Fernando handled and Unix programming and Hua loaded the graphics library into the project.

Software Environment:
For this project, we used Github for version control. All of our files are shared including references, documentation, and source files. Our project was written purely in OCaml and is not ported to any other languages.

To achieve graphical results, we utilize Graphics library that is pre-installed along with OCaml to render 2D objects on the screen to simulate our platform game. We also employed Thread library to delay the frame rate during the drawing process to simulate realistic object movements (otherwise the computer can compute their updated movements too fast to be pleasing to the eyes). We have chosen the delay rate to be 24 frames per second, the natural frame rate that humans can perceive smooth movement. Lastly, we used General Polygon Clipper (GPC) library to detect collision between two general polygonal shapes. We have implemented an additional method to provide proper boolean output that we can use for our own purpose.

Project Log:

Commits from GitHub:

commit 4bbd5f0f3c574887601057c47bd8438f510cc9fd
Author: Fernando Luo <luofernando@gmail.com>
Date: Wed Dec 19 11:35:37 2012 -0500

Final Report still need to do sec 4 and 7

commit bade550e85a170a4b7cccaba9e1ab29aeccf7125
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Wed Dec 19 10:55:48 2012 -0500

Fixed some bugs, started testing loop for game

commit 86293fb9fbfba1462fda6e85c839c51a1c961c35
Author: Fernando Luo <luofernando@gmail.com>
Date: Wed Dec 19 10:54:46 2012 -0500

key_pressed

commit 16a36c03043cd52e0b9c5b49086a57ab6797aa38
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Wed Dec 19 10:31:40 2012 -0500

Key

commit 9a2acb9935353cef26477a6345f8585fcc8101aa
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Wed Dec 19 10:31:30 2012 -0500

Key

commit 9384ac7413666a8f035898f454de7270c1273b1a
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Wed Dec 19 10:28:15 2012 -0500

Made some code cleanups in execute

commit 805da0a0cdde54f417df6e456f795d0f7b86adeb
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Wed Dec 19 10:25:31 2012 -0500

Fixed some typos

commit 4cab70f93c9748c887cadea170f28fc4f879876f
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Wed Dec 19 10:17:40 2012 -0500

Deleted compiled files, cleaned up execute

commit a037cf3f7c073b3decaa89d1f9475e27180ffd4d
Author: Papoj Thamjaroenporn <episer@gmail.com>

Date: Wed Dec 19 10:17:39 2012 -0500

    Clean

commit cb56b9d7b93d47e6112c140054deed7a32e689a6
Author: Lucy <peachie.monkey@gmail.com>
Date: Wed Dec 19 10:08:23 2012 -0500

    Made clean, added drops to prevent stack overflow

commit 25b2c3ba9010572d99207fd30aea8f5de7988e5f
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Wed Dec 19 10:11:04 2012 -0500

    More tests

commit 06f92e3d33faa6a23ec54e00e216a8c163d86575
Author: Fernando Luo <luofernando@gmail.com>
Date: Wed Dec 19 10:09:07 2012 -0500

    Final report sec 8

commit ad5267387d17121eaa3a5a32a6ef2912c6f3ee89
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Wed Dec 19 10:08:24 2012 -0500

    More codes in execute: checkCollision

commit c974f19ed4cc51ccac1221400af724720d3327c4
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Wed Dec 19 09:41:41 2012 -0500

    Cleaned up execute file and updated push1 test

commit a7413a633efbfa3d5b8480572064f2cdf806915b
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Wed Dec 19 09:32:14 2012 -0500

    More codes in execute: checkCollision

commit 20224c3359c5ac9fdc14a50d5127df3e046df5f4
Author: Lucy <peachie.monkey@gmail.com>
Date: Wed Dec 19 09:25:03 2012 -0500

    Add code to draw brick from stack !

commit c7b6ed3685c5b66591d29fb2290a70b441d0a770
Author: Lucy <peachie.monkey@gmail.com>
Date: Wed Dec 19 09:11:02 2012 -0500

    Trying to get brick data from stack

commit 2fed44e4cb7dfc29125f4afce6186591771335c0
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Wed Dec 19 09:11:45 2012 -0500

    Cleaned up more code

commit d02a0dfb28c1bec49598f686d363d37dbd2df9d3
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Wed Dec 19 09:00:24 2012 -0500

    Implemented Array count function and cleaned up code

commit 4cae46d810f36cfd964a6320a2eb61c82779b99a
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Wed Dec 19 08:33:45 2012 -0500

    Update collision and key

commit d9bde6c454a8fd2b588ad2c0a7fb28a968049b35
Author: Lucy <peachie.monkey@gmail.com>
Date: Wed Dec 19 08:29:32 2012 -0500

    Adding code to load bricks from stack into structs in execute

commit 083f0cdd47274d1ea676bf9ec61acea89fae6e27
Author: Fernando Luo <luofernando@gmail.com>
Date: Wed Dec 19 08:16:59 2012 -0500

    Final Report sec 5, 6, 8(testing code)

commit a6697c33141cb7798f38e8f1146bd16e02aa999d
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Wed Dec 19 06:34:34 2012 -0500

    Enforced binary operations on integers only

commit 61ded150d289f91d08d49ad14531dc822c07df61
Author: Lucy <peachie.monkey@gmail.com>
Date: Wed Dec 19 06:32:06 2012 -0500

    Added debugging code

commit 15815eaa54713ee9133899f7b384ffbd7eb6b989
Author: Lucy <peachie.monkey@gmail.com>
Date: Wed Dec 19 06:20:07 2012 -0500

    removed syntax error causing extra 'in'

commit 18a7a8498635832ecfeb751339bd5557e34f7806
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Wed Dec 19 06:19:54 2012 -0500

    Fixed type

commit 003df848b74818e5356bae67191eafe72bbb10ad
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Wed Dec 19 06:17:51 2012 -0500

    Fixed initializatino errors

commit ce30a8614719ce20083f583e00bb19d6550245cb
Author: Fernando Luo <luofernando@gmail.com>
Date: Wed Dec 19 06:17:20 2012 -0500

    Final report section 3

commit c6f98566ed6d82cb62ad210b58f393f4e0bd8ab7
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Wed Dec 19 06:14:38 2012 -0500

    Added intermediate files

commit 731d32ba312d98b73ae1531aeb68e2b875c0329d
Author: Lucy <peachie.monkey@gmail.com>
Date: Wed Dec 19 06:12:06 2012 -0500

    Delete

commit c3fefafd2d3fa7cb794aed7533714dee03fd4383
Author: Lucy <peachie.monkey@gmail.com>
Date: Wed Dec 19 06:11:45 2012 -0500

    Added player and block global variables to execute

commit 8a4ab33846ff7e073c5e32cccfad8a4597bdfdec

Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Wed Dec 19 05:02:52 2012 -0500

Implemented type checking for assignment

commit 4eba7019507327434db4fba5077f8fb65d22a77a
Author: Fernando Luo <luofernando@gmail.com>
Date: Wed Dec 19 04:54:26 2012 -0500

combined makefile Test_GraphicsGPC

commit 4bd9a350bfbe39c7d20ea1b911eb3170f057d41f
Author: Lucy <peachie.monkey@gmail.com>
Date: Wed Dec 19 04:48:39 2012 -0500

Draws the player; added if test; edited final report

commit aee3f21e64d72b35b35c1a7d681bd143a1c095fb
Author: Fernando Luo <luofernando@gmail.com>
Date: Wed Dec 19 04:44:22 2012 -0500

blocks wrap around

commit a2a7311417d5ac0bb789d6cbb85809ff4b67a8b5
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Wed Dec 19 04:33:16 2012 -0500

Now supports polygon collision detection

commit 67fc9fd02d134657b813d06511b2117a3d5ed0a6
Author: Fernando Luo <luofernando@gmail.com>
Date: Wed Dec 19 03:38:41 2012 -0500

make clean

commit eee07b976e933c43f93861f20f2990a03f7819aa
Author: Fernando Luo <luofernando@gmail.com>
Date: Wed Dec 19 03:38:32 2012 -0500

normal and reverse gravity (and bouncing!)

commit 313e4b55e5abf3d143120d8b88cd6a8bfe67a0c8
Author: Lucy <peachie.monkey@gmail.com>
Date: Wed Dec 19 02:49:02 2012 -0500

Fixed $run body off by one error, added code for "draw player"

Get $vertices from stack and convert to list of int

commit 63eb0fa8cdf4bc86ecb25fd9c9dc250dd320e440
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Wed Dec 19 02:05:48 2012 -0500

Supported drawing polygons in testGraphics

- Add several helper methods to allow polygon drawing, translating
relaitively and absolutely, and finding min/max.

commit f718255cd5b24edf0e203b6cfeec54bfcfc469ad
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Wed Dec 19 01:22:37 2012 -0500

Moved around environment table

commit 98ca351934cc1033c27a192946b51fd6d0238505
Author: Lucy <peachie.monkey@gmail.com>
Date: Wed Dec 19 00:45:09 2012 -0500

Updated reference code so that global variables can have local references and vice versa.

Edited tests too.

commit 8551c4bea0fae4e40fca695f6f437d05dc373d4b
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Wed Dec 19 00:37:23 2012 -0500

Added a Push built in function

commit ada7a6a519336454b94253551f26feadb3951f4d
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Wed Dec 19 00:11:49 2012 -0500

polygon fill test

commit 4febb52d1886f66ae186fa92a88a370f6b191aba
Author: Fernando Luo <luofernando@gmail.com>
Date: Wed Dec 19 00:10:03 2012 -0500

Final Report

commit df8afe15d479e7f14b2cf7af4e1a5e6fe32cbc32
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Tue Dec 18 23:32:26 2012 -0500

Add new test graphics: now supporting polygons

commit ef645517ebe66fcff8e3ac0f496022e777fa076c
Author: Fernando Luo <luofernando@gmail.com>
Date: Tue Dec 18 22:48:37 2012 -0500

Rts for all types, test for all types and arrays

commit b7b5367aa66cbbaea083ba0c1a424eb9b1b70931
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Tue Dec 18 22:19:02 2012 -0500

change graphics and other small things

commit fdd4c6b9446a51c719ec1e0fe7a80b33b8430a52
Author: Lucy <peachie.monkey@gmail.com>
Date: Tue Dec 18 22:08:43 2012 -0500

Added basic game to test. Edited "Run" function in compile

commit 98e0ad6128ddcf1503272627efdbe84e7330db3c
Author: Fernando Luo <luofernando@gmail.com>
Date: Tue Dec 18 21:42:26 2012 -0500

make clean

commit 34c7cf1c0fdc0d546968d5d4f06d74995a078253
Author: Fernando Luo <luofernando@gmail.com>
Date: Tue Dec 18 21:42:16 2012 -0500

Rts string, test string return, make clean

commit 99067d5f988100954b328eefc0fe7fe2334c2f2d
Author: Fernando Luo <luofernando@gmail.com>
Date: Tue Dec 18 21:32:43 2012 -0500

tested arraybrick and arrayplayer return

commit a6968c36e893766d433aedcdee83da7032a03e20
Author: Papoj Thamjaroenporn <episer@gmail.com>

Date: Tue Dec 18 21:26:58 2012 -0500

    More updates for FInal Report

commit 89eb3cf3f21bb0873ff21c92934bacb5a830b800
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Tue Dec 18 21:15:42 2012 -0500

    Add test-array7.rc

commit fb44f2bb15cdafc57f612fc0251d7b4596eb83bb
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Tue Dec 18 21:06:55 2012 -0500

    More progress on Final Report

commit dbf7ca357b58276a040f795ad863d8a496180106
Author: Lucy <peachie.monkey@gmail.com>
Date: Tue Dec 18 20:50:38 2012 -0500

    Fixed error in assigning references to .$vertices

commit fd8cfe003d329c075c74e05c95d8ffed0593f184
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Tue Dec 18 20:50:24 2012 -0500

    Updated error dialog.

commit 59464596595707ddd2b459c1803b60f1021816ac
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Tue Dec 18 20:43:07 2012 -0500

    Added catch to execution to print pc at point of failure

commit edb251e4d38ea0e2c14dfd281a004d6422c14aa3
Author: Fernando Luo <luofernando@gmail.com>
Date: Tue Dec 18 20:41:44 2012 -0500

    Rts type 6-10, tested arrayint

commit ab2a47248c2a3ec30f6795e4b120d80e940d65b3
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Tue Dec 18 20:22:33 2012 -0500

    Updated execute with built in function to count array

commit 2c5d980cd2a67b6c3f6e752e71435d42dedac73e
Author: Lucy <peachie.monkey@gmail.com>
Date: Tue Dec 18 20:16:06 2012 -0500

    Add map tests, edit make so arrays initialize with all
zeros

    Also edited brick tests

commit 9c751c6edd5ff935699bb440aad0b4674671325e
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Tue Dec 18 19:31:51 2012 -0500

    Add final report document file

commit 54f2b772c06edaacb39bc29a940a8d61e9a0d16f
Author: Fernando Luo <luofernando@gmail.com>
Date: Tue Dec 18 19:23:47 2012 -0500

    Creating and retrieving bricks in an array

commit 3d206cea223727640e82396a0352de3363e7703a
Author: Fernando Luo <luofernando@gmail.com>
Date: Tue Dec 18 18:42:24 2012 -0500

    array of players supported, test files generated

commit f72e334607c536e5dab7f2d69b5c0e39a7312187
Author: Fernando Luo <luofernando@gmail.com>
Date: Tue Dec 18 18:22:04 2012 -0500

    test-brick, test-player

commit 35559ee612694aae33e1c93c0af3517b9efac4d9
Author: Fernando Luo <luofernando@gmail.com>
Date: Tue Dec 18 18:15:16 2012 -0500

    tests.ml, fixed Player() in compile, added player test

commit d4dfa30a9f8145e479c349e746bf8f8d2c883c8b
Author: Fernando Luo <luofernando@gmail.com>
Date: Tue Dec 18 17:18:40 2012 -0500

    test-obj files

commit 50641dd4d9c2c919e897064a8298771b8cfc8314
Author: Lucy <peachie.monkey@gmail.com>
Date: Tue Dec 18 17:06:42 2012 -0500

    Added code to support access/assignment by reference;
updated execute

    Also added tests for functionality by reference. Updated
execute so
    that accessing/assigning to arrays will "Drp" the values
pushed onto
    the stack (array address & index).

commit a14216011bb47bf32b8b609473feb62533c936e8
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Tue Dec 18 16:29:28 2012 -0500

    Clean testall.sh

commit ad01bee3187319a945acc3a976eb0bddf3e3ce49
Author: Fernando Luo <luofernando@gmail.com>
Date: Tue Dec 18 16:16:42 2012 -0500

    clean .diff files

commit cb8e2613ded429e275ac85b141f3869f1f1d6c21
Author: Fernando Luo <luofernando@gmail.com>
Date: Tue Dec 18 16:16:21 2012 -0500

    clean, and testall getopts

commit cf1668eec6531ea60073eee08e694b3952fc433e
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Tue Dec 18 16:04:51 2012 -0500

    Worked on shell script a bit more + Makefile

commit 37e430ddd8604af981563a51fbdac2bf8407f22e
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Tue Dec 18 15:26:56 2012 -0500

    Added Shell Script and Test suite reference

    - Now shell script can run against reference to automate
test suite
    checking

commit 41f85d3a42e742fbd3cbfdfd17e6af75f85e82bc
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Tue Dec 18 14:30:11 2012 -0500

Clean stuff

commit 8176b41a29be9009fc9919389313479a4ac85c67
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Tue Dec 18 14:22:50 2012 -0500

clean intermediate files

commit f3e07361983d2f11c778d7454c370f2b40b66126
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Tue Dec 18 14:21:37 2012 -0500

Added new AST expressions to support $brick.$array[$index]

Also, added new shell script to autmate test cases

commit 3dd2a7fa536091e6e516eaf0bfb3150dc7e03762
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Tue Dec 18 12:49:26 2012 -0500

Small increments to compile and test cases

commit 75eac21002d1cfff43af31fd4ffbfb41cc861572
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Tue Dec 18 05:10:13 2012 -0500

Small increments to compile and test1

commit 80ccab0c120641d7b3dbf45383ab7d916058390f
Author: Lucy <peachie.monkey@gmail.com>
Date: Tue Dec 18 05:03:22 2012 -0500

Fixed array errors; added array tests.

Fixed access and assignment for local and global arrays. Also added 5
array tests. test-array5 demonstrates how to use a global array to
'return' arrays from functions

commit 1e7ed4e737726fd57ae73e7ba9bab6b0fe4f7551
Author: Lucy <peachie.monkey@gmail.com>
Date: Tue Dec 18 01:56:41 2012 -0500

Updated two tests

commit 739adbf909690b55a28a330a2c1869b70c96a14f
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Tue Dec 18 01:55:19 2012 -0500

Created new tests + updated Array for compiler and execute

commit fb16161724c97b5b98bdea22f59cb358c6c19a02
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Tue Dec 18 00:32:19 2012 -0500

Add testGraphics to support collision

commit eb0537498dbdb658d88b6eea735126f73bdec4f1
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Mon Dec 17 23:14:27 2012 -0500

Fixed test-var1

commit 599e2fc030ff1cead9f763126f7205f27b23c8ae
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Mon Dec 17 23:13:19 2012 -0500

Edited tests, some are working

commit c30f7ee03e4c7d9d62fa7c1ce68129061aeaa2c8
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Mon Dec 17 22:54:06 2012 -0500

Fixed compiler and execute bugs

commit 65a567190fc4171cb431ca400a413cf65c2bff25
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Mon Dec 17 22:32:05 2012 -0500

Fixing compiler errors

commit 3ca058cd109cc2da5bff3779788a54af687da292
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Mon Dec 17 22:19:19 2012 -0500

Changed Not to accept expressions

commit c5e12a78b955496e0097c06dc228d1c7d2f49408
Merge: cb77728 315f9b2
Author: Lucy <peachie.monkey@gmail.com>
Date: Mon Dec 17 22:18:33 2012 -0500

Merge branch 'CompileTest' of https://github.com/klin01/PLT into CompileTest

commit cb77728241bb6bb8b53defd39da8de5090e86d38
Merge: 0540f21 fcdaa2f
Author: Lucy <peachie.monkey@gmail.com>
Date: Mon Dec 17 22:18:31 2012 -0500

Fixed compile syntax errors

commit 315f9b24da3b2fbf112b7f552dab32b3dcc578ca
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Mon Dec 17 22:12:07 2012 -0500

Fixed lfpa

commit 6fc96c66269f51bc65a1656497491cc552632b6d
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Mon Dec 17 22:09:02 2012 -0500

Fixed some syntax errors in compiler

commit 0540f217dbf2eda7c8c908ea3118141667a38362
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Mon Dec 17 22:07:34 2012 -0500

Removed some unused bytecodes

commit fcdaa2faa6623b4c532a8ae4764d605dcc71d1df
Author: Lucy <peachie.monkey@gmail.com>
Date: Mon Dec 17 22:05:15 2012 -0500

Fixed syntax errors in compile

commit 43af9596fa86957e563113d708e8f521ba647321
Author: Fernando Luo <luofernando@gmail.com>
Date: Mon Dec 17 22:04:16 2012 -0500

Return with only int

commit 61d0a66b56a83fe56ba67d0402f0741c7e963b48
Merge: e9823e9 7765230
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Mon Dec 17 21:59:54 2012 -0500

Merged StartFromMicroC to CompileTest

commit 776523089d1363e84563eba1bf56b50739e30c6c
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Mon Dec 17 21:39:59 2012 -0500

Updated variable sizing on compile and execute

commit e9823e9f587eeb8b6464ff4663958e5b663ec3ca
Author: Fernando Luo <luofernando@gmail.com>
Date: Mon Dec 17 21:47:20 2012 -0500

execute.ml - for and while loop

commit cb3534f97dce2512e0cda55bbdebbb33aa983143
Author: Fernando Luo <luofernando@gmail.com>
Date: Mon Dec 17 21:40:01 2012 -0500

bytecode and test files

commit 0c51a75175c1b7cfa4a64e3d274a4ab028991305
Author: Lucy <peachie.monkey@gmail.com>
Date: Mon Dec 17 21:33:13 2012 -0500

Updated ast, bytecode, compile, execute, parser

Parser now passes color as 3 ints into brick and player constructor;
updated execute pointer offsets; updated compile code (brick and player
constructor, general debugging); removed parameter for array load/store
in bytecode and updated ast parameters.

commit b82bf379a05e780b97b1b01ac9b9d12c4b7cfc9c
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Mon Dec 17 20:02:32 2012 -0500

Delete cmo files

commit b37fe2d63a141cd57973f3b4a519d14b16e85b0b
Author: Fernando Luo <luofernando@gmail.com>
Date: Mon Dec 17 19:58:36 2012 -0500

Changed compile.ml

commit
d85b2b9121d7e4503a0189ddaddee1a3290330a7
Author: Lucy <peachie.monkey@gmail.com>
Date: Mon Dec 17 19:53:25 2012 -0500

Updated Lfpa and Sfpa

commit 648f84822a698660dff6e487bc01c1e8677e06e2
Author: Fernando Luo <luofernando@gmail.com>
Date: Mon Dec 17 19:50:25 2012 -0500

Changed and commented out code

commit 02cf92d4bf332aa27261a0fd59e0d6037bd9dce4
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Mon Dec 17 19:49:02 2012 -0500

Started on handling Run command in compiler

commit c6c3acc1e041e30a6a58916d7ab5d7d8c7021902
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Mon Dec 17 17:54:46 2012 -0500

Modfiy REF

commit 4ce064a81f653d5b2e7ecc7614e45f5ef92d2aff
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Mon Dec 17 17:52:08 2012 -0500

Make clean

- Changed some AST syntax from expr to string, making the syntax more
specific.

commit 87d7677fb5a72dabddcb0e8666029243bc6fd0b9
Author: Fernando Luo <luofernando@gmail.com>
Date: Mon Dec 17 17:15:06 2012 -0500

arithmetic in execute.ml

lod and str

commit dd38519648199d6741cce7aa196e7ee4bfed44dc
Author: Lucy <peachie.monkey@gmail.com>
Date: Mon Dec 17 17:02:25 2012 -0500

Edited compiler referencing and array access / assign

Array instructions will push array address and index onto stack and
then take values to find correct index address (instead of passing the
array address via Ldfa Loda Stra and Sfpa)

commit 81623982fe08c15ffab71d05b485a5302f2db989
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Mon Dec 17 16:59:57 2012 -0500

Added array to string in tests file

commit 556a3414c1355bca9f567f38f033e282c69a9d7a
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Mon Dec 17 16:58:53 2012 -0500

Update Compile.ml

commit 2a4842376979d28d1ec444b032bbfcac6f7b8f5f
Author: Lucy <peachie.monkey@gmail.com>
Date: Mon Dec 17 16:57:58 2012 -0500

Edited types for array and function references to string

commit 71b31133adcf1b9ab386ecc9f66652faf9b5b369
Author: Fernando Luo <luofernando@gmail.com>
Date: Mon Dec 17 16:44:34 2012 -0500

Arithmetic in str (executeml)

commit b9f83886593d79b3be8f50c7a9b2e7c7e0b0077a
Author: Fernando Luo <luofernando@gmail.com>
Date: Mon Dec 17 16:26:07 2012 -0500

MakeB, MakeM, and MakeP in execute.ml

commit 6fd7b745cb4fbd4a59a8b89d5ffcbec2bde0063e

Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Mon Dec 17 16:13:50 2012 -0500

Fixed compiler errors and scanners REF tokens

commit 0f000b1670a3102879507c29bf85d87cabff8c5a
Author: Lucy <peachie.monkey@gmail.com>
Date: Mon Dec 17 15:13:24 2012 -0500

Making changes to compiler and parser to support '.' referencing

commit bdd69484ee6406870ba9649034b021394357707b
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Mon Dec 17 12:44:44 2012 -0500

Makefile updated:

Use make to generate compiler code.
Use make runtests to run the AST tests as usual

commit f625b9fe9c3d07a11f41d88729a4a355043ccfdf
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Mon Dec 17 10:34:44 2012 -0500

Committing updates to ast/parser and a lot of code for execute

I've written up most of the execute code for stores and loads

commit 25d828058d1451e8fddeb16e7e3c0d4af162b9ac
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Mon Dec 17 03:42:10 2012 -0500

Changed color representation from string to 3 ints

commit 97b2851af2850644d2f33ab2d4112a54b5c7b53a
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Mon Dec 17 02:26:57 2012 -0500

Changed ast.mli to ast.ml

commit 9b23d884dd7ed19f440446c9ff4f9bc1d06e49b7
Author: Lucy <peachie.monkey@gmail.com>
Date: Sun Dec 16 23:08:37 2012 -0500

Matched more bytecode instructions to stack instructions in execute

commit 1c9c19f8c2134c3b0d3eb095e9938d30a0ea0594
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Sun Dec 16 23:08:17 2012 -0500

Updated compile file

References fixed, still need testing

commit 1bc8abd0b028c42199e6710fdb1d7e71e6749c4c
Author: Lucy <peachie.monkey@gmail.com>
Date: Sun Dec 16 22:40:26 2012 -0500

Started editing execute to match bytecode and compiler

commit 1234eb1994f560f62bc4f437078165b4b6f96b03
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Sun Dec 16 22:36:39 2012 -0500

Add test graphics 2: keyboard input + player block + moving obstacle

commit da3e2cfa85f0a87a81772621b64928cca76170bf
Author: Lucy <peachie.monkey@gmail.com>
Date: Sun Dec 16 21:08:22 2012 -0500

Added braces around 'match' in Not

commit 9c335fb11509a94fc22e26dcce38cb10931bf73f
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Sun Dec 16 21:07:47 2012 -0500

Updated Compile.ml

Made player/brick/map type checking more explicit

commit ddcc626160f66e3ffd4b838900a1cdd93d09830e
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Sun Dec 16 20:09:40 2012 -0500

Changed shape/height/width attributes of an object

Those attributes are now represented by an array of points

commit 901c4690edd906c2bfccf2f6d904c42efae8c2b4
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Sun Dec 16 18:04:36 2012 -0500

Removed Array token from AST

commit 3858d5a0b4678b99dfbc6840e958b9d47dc595cd
Author: Lucy <peachie.monkey@gmail.com>
Date: Sun Dec 16 18:01:56 2012 -0500

Completed enum and expr functions of compile

commit 4d5880458c8c9c5006c587799d7c7431f7cf1388
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Sun Dec 16 16:31:45 2012 -0500

Removed git text

commit deb0101f20680583a12c0df87c5fd4ca4f05e205
Author: Lucy <peachie.monkey@gmail.com>
Date: Sun Dec 16 16:28:21 2012 -0500

Added to compiler, still missing commands for Array

commit 1ccde0c37263fabcfcb7dfe5eabfa4aac7e1f473
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Sun Dec 16 16:28:37 2012 -0500

Cleaned up AST/Parser/Scanner

commit 1b26c93561da58e918dc4c8ce25087b432402265
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Sun Dec 16 15:41:06 2012 -0500

Removed Invocation of functions

commit 3c7681b227cda2d4845db9e9359e256d6cf170b6
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Sun Dec 16 04:31:54 2012 -0500

Delete some temp files

commit 7507f85d46740a870d008f6cb34192f13f1a7e0f

Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Sun Dec 16 04:31:35 2012 -0500

Update testGraphics to have falling animation!

commit dca80cf7fecee3b1ff5d73348368690753449fe5
Author: Lucy <peachie.monkey@gmail.com>
Date: Sun Dec 16 03:47:20 2012 -0500

Removed extra error causing text in parser

commit a2628dd17748a239a8530eefb2d36667d208ad5e
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Sun Dec 16 03:44:26 2012 -0500

Add test_graphics

Demonstrating how Graphcis package works. Combined with Thread to allow
UI to stay for a length of time.

commit 45e43a2ded57c5047b41eae1ae06b31690b14e83
Author: Lucy <peachie.monkey@gmail.com>
Date: Sun Dec 16 03:41:45 2012 -0500

Edited enum function (and made clean)

commit 82958b07fc10c1d5291098a0120e4736a9364459
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Sat Dec 15 20:44:15 2012 -0500

Add TODOs list for each ml files

commit d84c5ec03d38293de6c477f4fe71a0af6b7abf06
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Sat Dec 15 20:38:45 2012 -0500

Add TODOs

commit 711517e996fa0690f5438e3dafad70fe76a50427
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Sat Dec 15 20:37:17 2012 -0500

Added Array Access to parser and scanner

Also added negative integer support

commit 714d7e244373c17ec1dca940e68414438aa5342e
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Sat Dec 15 20:33:56 2012 -0500

Edit test codes to comply with Retro syntax

commit 43659b9fc2fd51a9b66f74874b33ad7e7481faa0
Author: Fernando Luo <luofernando@gmail.com>
Date: Sat Dec 15 20:33:09 2012 -0500

added to bytecode, and compile.ml

commit 1d2b633773fd37146463c10daa9ca26509fd8e6e
Author: Fernando Luo <luofernando@gmail.com>
Date: Sat Dec 15 18:30:52 2012 -0500

bytecode.ml and execute.ml

commit 9b5e740e92e900723e437a2ff92b34de82b73658
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Sat Dec 15 18:27:26 2012 -0500

Removed unwanted tokens from scanner and parser

removed Height/X/Y/Width/Generator

commit 980c3416b79a460874ccdebb75a56c72418878ce
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Sat Dec 15 18:26:22 2012 -0500

Removed LiteralBool/LiteralChar/LiteralFloat

commit 6c0639f82b8e80a87cde708415f3226abcda07d2
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Sat Dec 15 18:04:07 2012 -0500

Updated Parser and Scanner

Removed RUN as keyword, updated test file as well

commit e554cbd5ab4894ed3e29e50a9c6dd7246d66d1ed
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Sat Dec 15 16:59:25 2012 -0500

Updated parser and runtests file

commit 05391f3cbc4da274138d6b2a15788b296f1ace22
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Sat Dec 15 00:11:23 2012 -0500

Got a working parser and test code

commit 9e352fcfd4543720ee7acd5c4e9b3c3ddd91302c
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Fri Dec 14 21:41:47 2012 -0500

Updated scanner/parser/ast, added test files

commit 838b1adbfed51085ed7034dc051d8a1faba2d0f1
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Fri Dec 14 21:02:32 2012 -0500

Updated parser,scanner,ast

commit 95d8aeee52063b3ffc5bf1b3f284c9d2ff4fd3ed
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Fri Dec 14 17:19:48 2012 -0500

Additional Changes

commit 986b06aac15c07f4cabbc81c2710b54678832cfd
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Fri Dec 14 17:12:22 2012 -0500

Added sample code

commit 14ea1b373c2308a4af53725866d065964c5c441b
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Fri Dec 14 17:12:07 2012 -0500

Modify Parser and Scanner to add new Types

commit 670466eb99241108e4bfd6e472e85f3212103215
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Thu Dec 13 18:29:17 2012 -0500

Add compiling instruction and modify Makefile clean

commit 49ede45631729429d09472b4b4e819319af08051
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Thu Dec 13 17:15:26 2012 -0500

Add new branch: Start From MicroC

Build up the code from MicroC framework

Will add stuff from other branches as well.

commit 5a22a2d5ed1b637bdd6fff26f516deec195d819f
Author: Lucy <peachie.monkey@gmail.com>
Date: Thu Dec 6 20:14:09 2012 -0500

Removed unnecessary quote code

commit bf89bf166a0512f7893549a374befd90c775731b
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Mon Dec 3 02:26:18 2012 -0500

Figured out how to compile ocaml code with OpenGL binding

Figured out how to compile ocaml code with OpenGL binding

See instruction file for installing and compiling LablGL.

OpenGL_OCaml_Instruction.txt

commit 3770c55907ddfd02263e2064afc7a1e7645743b3
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Sun Dec 2 21:06:45 2012 -0500

Add OpenGL testfile and instruction on how to install Lablgl

commit 1ead4c5a092527899259313ac6b63467be5dc617
Author: Lucy <peachie.monkey@gmail.com>
Date: Sun Dec 2 21:03:41 2012 -0500

Edited singlequote and doublequote

commit 92f610e495a99e399dbe2333c9a2ec79f65cd751
Author: Lucy <peachie.monkey@gmail.com>
Date: Sun Dec 2 20:25:51 2012 -0500

Added float and char to scanner

commit b8c2ca9bc4ac6d55185e9534bf174129661ed186
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Sun Dec 2 17:27:13 2012 -0500

Good stuff

commit a156f01919ddfe72f87001eb000810a7e9233d5f
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Sun Dec 2 15:05:46 2012 -0500

Made a lot of changes gl

commit 83c92f0c72524994a72cdefbcd635ded44891051
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Sat Dec 1 20:55:40 2012 -0500

Random test file, doesnt work

commit 65d7474c948583c928112e85aa6e1671fa9b75d7
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Sat Dec 1 20:53:54 2012 -0500

Add useful readings: Ocamlyacc and Ocamllex

commit 0591cf2279319ff8633a9e66bce5e51dbcc159c4
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Sat Dec 1 19:57:01 2012 -0500

More code

commit 45bb4a38b9157641365481445fc7324e7141a1e6
Author: Lucy <peachie.monkey@gmail.com>
Date: Sat Dec 1 17:20:30 2012 -0500

Added ocamlyacc tutorial

commit 793cee26749f6797587540a6d7a10f7cd36bbfe7
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Sat Dec 1 17:16:33 2012 -0500

Added parser and ast

commit 061368beeb6406d9f74bbe2d817c22505fbb4b83
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Sat Dec 1 17:18:15 2012 -0500

Revert "Clean my branch."

This reverts commit 3a900894d78f107937f79bc0b65ea1aae60aebd3.

commit 3a900894d78f107937f79bc0b65ea1aae60aebd3
Author: Papoj Thamjaroenporn <episer@gmail.com>
Date: Sat Dec 1 17:14:21 2012 -0500

Clean my branch.

commit 9e23acb5c39ca5c929b7ae7918127331e4bc508e
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Sat Dec 1 15:23:52 2012 -0500

Fixed single quotes

commit 2be908714fd1e3874c0324d334849d818cf38ce4
Author: Fernando Luo <luofernando@gmail.com>
Date: Sun Nov 18 17:46:35 2012 -0500

Started Scanner.mll, still need to implement rules for quotations

commit 82ee89c61428c08306abe825f8633cfdcb21158a
Author: Kevin Lin <lin.kevin.01@gmail.com>
Date: Sun Sep 16 00:35:32 2012 -0400

Initial commit

5. Architecture Design

5.1 Parser/Scanner

Inspired by the MicroC compiler, RetroCraft utilizes the Scanner in conjunction with the Parser to read the program and generate the abstract syntax tree of the program. The scanner file first converts the source code into discrete tokens. Rules in the scanner file allows for multiline and single line comments. We've also identified all the reserved keywords (**Section 3**) as tokens to prevent users from mistakenly use them as variables. Furthermore, our scanner guarantees that all identifiers start with ($).

The parser invokes the program routine to generate a list of variable declaration and a list of function declaration. This architecture satisfies our language due to the presence of global variables.

```
function $main : () {
     int $i;
     $i : 0;
     $printint ($i);
}
```

would be translated to:

```
FUNC ID ASSIGN LPAREN formals_opt RPAREN LBRACE
     INT ID SEMI
     ID ASSIGN (expr-> LITERALINT) SEMI
     ID LPARENT actuals_opt RPAREN SEMI
RBRACE
```

Next it would be parsed to:

```
{ fname = "$main";
 formals = ();
 locals = $i;
 Body = Assign ($i, 0);
 @ Call ("printint", $i)
}
```

Then finally into bytecode:

0 OpenWin
1 Jsr 3
2 Hlt
3 Ent 2
4 Init 1 2 1
5 Litint 0
6 Sfp 2
7 Drp
8 Lfp 2
9 Jsr -3
10 Drp
11 Litint 0
12 Rts 0

To be executed:

$ ./retrocraft < test/test.rc
0

5.1.2. AST

The AST first enumerates the tokens and specify and associativity between operators to reflect standards such as PEMDAS. The abstract syntax tree primarily defines the core structure of a retrocraft program. The parser will reference this file in order to generate an tree.

5.1.3. Bytecode

Our bytecodes are as follows:

```
Litint of int          (* Push a int literal *)
Litstr of string       (* Push a string literal *)
Drp                    (* Discard a value *)
Bin of Ast.op          (* Perform arithmetic on top of stack *)
Lod of int             (* Fetch global variable *)
Str of int             (* Store global variable *)
Loda                   (* Load global array variable *)
Stra                   (* Stores global array variable *)
Lfp of int             (* Load frame pointer relative *)
Sfp of int             (* Store frame pointer relative *)
Lfpa             (* Index is evaluated and put on top of stack*)
Sfpa                   (* Stores frame pointer of array *)
Lref             (* Loads a value onto the stack from an address
*)
Sref           (* Saves a value from the stack into an address *)
Jsr of int             (* Call function by absolute address *)
Ent of int             (* Push FP, FP -> SP, SP += i *)
Rts of int     (* Restore FP, SP, consume formals, push result *)
```

```
Beq of int          (* Branch relative if top-of-stack is zero *)
Bne of int      (* Branch relative if top-of-stack is non-zero *)
Bra of int              (* Branch relative *)
Make of int             (* Shift stack pointer by 1 for Player,
Map, Brick; Adds vartype_id to first space in arrays *)
Init of int * int * int
PrintScore              (* Prints the user's current score on the
top left *)
Hlt                     (* Terminate *)
Nt                      (* Negate 1 or 0 on top of stack *)
```

5.1.4 Execute.ml

Due to our language having more types than MicroC, we needed to differentiate our stack values from each other with an int typeID. The execute will read the bytecode, allocate a stack, and perform stack operations based on the program. Execute.ml maintains stack, frame and program pointers. Execute is also responsible for opening graphics window and performing object translations graphically due to close proximity to the actual data.

5.1.5 retrocraft.ml

This is the command line program that allows user to output the bytecode of the program instead of compiling. It traces through each command, displaying any pertinent information regarding stack operations, which makes it ideal for debugging.

6. Test Plan

To demonstrate the power of our language, we created various test cases to see the limit of our language. Retrocraft can handle from basic arithmetic to even slightly more complex math that employs recursion. (Fibonacci's series).

Retrocraft has great supports for `while`, `for` loops while endured numerous testing of `if` and `else` logic. Our language allows referencing of ids and also supports returning of all data types.
We further tested Array support by combining the Arrays with various data types and looping logic. Furthermore, we have included an automated testing script which will compare the output of each file to the supposed output of the test programs (`testall.sh`).

7. Lessons Learned

7.1 Papoj Thamjaroenporn

A lot of the times we have spent for this project have been invested toward the project proposal and language reference manual. As a group, we believed that if we carefully design the language and prospective features early on we would be best prepared to finish this project flawlessly. As it turned out, we ran into countless number of small technical problems that we had to solve and fix along the way just to get the basic Abstract Syntax Tree, Scanner, and Parser alone to work. I learned that the best way to tackle a big project that I have little related background is not to have the proposal as detailed and well-defined as possible, but to get my hands dirty as fast as possible. As I became more familiar with OCaml, and the architecture of a language compiler, I felt that I had a much better sense of estimation of how much I could achieve as a semester-long project Consequently, we modified our language features significantly to correspond with our potential. I would suggest to the PLT group in the future to rather get their hands early as fast as possible rather than trying to be precise with their proposals and reference manuals, since they can potentially change drastically over time. Although our project has not been as rich as we expected since the beginning, I am still proud of how much we have learned and accomplished during such a short time period.

7.2 Kevin Lin

Designing and building a programming language from the stack up was a deceptively difficult challenge. Beginning from the naive stages of brainstorming and wishful thinking, the true challenges we were going to face in the months to come were far from our minds. Breezing through the development of the scanner, parser and AST didn't help us come to terms with the nightmare of debugging and testing ahead of us. As such, we ultimately ended up wishing we had more time. Personally, I didn't realize how difficult or how long it would take to understand the subtle nuances of the development of the byte code and the management of the stack. After several deca-hours spent pouring over byte code output and stack traces to see why our Arrays weren't filling in the right indices, and correcting counting errors, a stronger understanding of the logic driving the system finally started to set in. But by then, much of the more naive decisions we had made earlier in the development process were starting to bite us in the butt. Given an infinite amount of time and stamina, we could have easily hammered out the kinks that came up because of inexperienced design but because of the lack of it, we were forced to settle for some bandaid solutions. Some of the bigger issues we were forced to go back and apply deep

fixes for, such as our short-lived plan to allow for the storage of both references and values. For the most part, I just wished we had spent more time on the design and planning part of the project, and as always, I wished there was more time to actually apply these lessons learned.

7.3 Lucy He

Among the many things I learned from this project, a key take away was an appreciation for functional programming and OCaml. Most used to coding in Java, I first thought OCaml was unnecessarily complicated. As the semester progressed and we developed our programming language, I quickly realized the great potential and versatility of OCaml, especially for writing a compiler. In retrospect, I am very glad that we were required to learn this new language.

Despite the conveniences provided by OCaml, I found this project very challenging. Building a compiler is not much like any other programming assignment I've encountered. It required us to deconstruct many things we've learned previously and think critically about ideas and conventions we use everyday. For that reason, I thought it was an extremely valuable learning experience. I definitely found the project extremely overwhelming at first. It was hard to get started when trying to fully understand the many components of a compiler. However, the challenge made it very exciting when I was finally able to follow the flow of data through our code as it all fell into place.

While I feel like I learned a lot, I was hoping we would finish with a slightly different final product. Our team's original plan was to design a language that simplified the design process for a slightly different style of computer game. Unfortunately, many of the challenges we faced did not become evident until we were already fairly invested in our code. It was a great challenge to continuously update the abstract syntax tree, bytecode interpreter, etc. so that they were consistent and functioning correctly.However – despite the challenges and in light of all the lessons – as we finish up this project, I know that it was a very worthwhile experience!

7.3 Fernando Luo

Majority of the project was actually deciding the structure and flow of our language. Unfortunately due to limited time and our inexperience with function programming, we over estimated what we could do in one semester. Originally, we intended to create a 2D platformer game akin to Super Mario Bros. Thought such as infinite scrolling, gravity, and other features came to mind. However, the largest obstacle for this project, I think personally, would be determining the AST and Parser for our language. Although these two are technically the most straightforward, it was the source of a lot of feature revisions and removals. My advice for future teams is to know the semantics of your language before diving into development. We had to learn the hard way that having to go back and change a bulk of the program due to one seemingly small change.

Overall, I benefited immensely from working with functional programming for the very first time. The thinking and developing process are very different from that of

procedural languages. Furthermore, I understand programming language translator across the entire stack, especially after we decided to use byte code to translate our program. Perhaps the most enjoyable part comes from us programming in Retrocraft to create our own game maps.

8. Sample Code

Appendix I: recursion

Fibonacci's Series:

```
function $fib : int (int $x)
{
  if ($x < 2) return 1;
  return $fib($x - 1) + $fib($x - 2);
}

function $main : void ()
{
	$printstring("Should be 1");
	$printint( $fib(0) );

	$printstring("Should be 1");
	$printint( $fib(1) );

	$printstring("Should be 2");
	$printint( $fib(2) );

	$printstring("Should be 3");
	$printint( $fib(3) );

	$printstring("Should be 5");
	$printint( $fib(4) );

	$printstring("Should be 8");
	$printint( $fib(5) );
}
```

Appendix II: Control Flow:

While Loop:

```
function $gcd : int (int $a, int $b) {

	while ($a != $b)
	{
		if ($a > $b)
			$a -: $b;
		else
			$b -: $a;
	}
	return $a;
}

function $main : void ()
{
	$printstring("Should print 2, 3, and 11");
	$printint( $gcd(2,14) );
	$printint( $gcd(3,15) );
	$printint( $gcd(99,121) );
}
```

For Loop:

```
function $main : void ()
{
int $i;
 $printstring("start");
 $printstring("Should print 1 to 4");
 for ($i : 0 ; $i < 5 ; $i +: 1) {
  $printint( $i );
 }
 //$printstring("end");


 $printstring("Should print 5 to 9");
 for ($i : 5 ; $i < 10 ; $i +: 1) {
  $printint( $i );
 }
}
```

If, Else, Else If

```
function $main : void ()
{
 if (false){
        $printint(42);
 } else if (true) {
        $printint(8);
 } else
        $printint(17);
}
```

Appendix III: Data Object and Arrays

Test-array.rc

This test case demonstrate various uses of array of integers: function calls with array of integers return type, array random access, and local array defined within function context. The source code should print consecutive number running from 0 to 4, then 0 to 14 respectively.

```
function $retIntArray : ()
{
      Array int $retArray;
      int $i;

      $retArray : new Array int;

      for ($i :0; $i < 5; $i +: 1){
            $retArray[$i] : $i;
      }
      return $retArray;
}


function $retIntArray2 : ()
{
      Array int $retArray;
      int $i;

      $retArray : new Array int;

      for ($i :0; $i < 15; $i +: 1){
            $retArray[$i] : $i;
      }
      return $retArray;
}

function $main : ()
{
      Array int $localArray;
      int $i;
```

```
	$localArray: $retIntArray();

	$printstring("printing returned array");

	for ($i : 0; $i < 5; $i +: 1) {
		$printint($localArray[$i]);
	}

	$localArray: $retIntArray2();
		$printstring("printing 2nd returned array");

	for ($i : 0; $i < 15; $i +: 1) {
		$printint($localArray[$i]);
	}

}
	$localArray: $retIntArray();

	$printstring("printing returned array");

	for ($i : 0; $i < 5; $i +: 1) {
		$printint($localArray[$i]);
	}

	$localArray: $retIntArray2();
		$printstring("printing 2nd returned array");

	for ($i : 0; $i < 15; $i +: 1) {
		$printint($localArray[$i]);
	}

}
```

Test-map3.rc

This test case demonstrate various uses of array of integers: function calls with array of integers return type, array random access, and local array defined within function context. The source code should print consecutive number running from 0 to 4, then 0 to 14 respectively.

```
Array int $vertices;
Brick $b1;
Brick $b;

function $retBrickArray : ()
{
	Array Brick $retArray;
	int $i;
	int $j;
	int $k;

	$vertices : new Array int;
	$vertices[0] : 300; $vertices[1] : 50;
	$vertices[2] : 300; $vertices[3] : 100;
	$vertices[4] : 250; $vertices[5] : 100;
	$j : -1;
	$k : 0;
```

```
	$retArray : new Array Brick;


	for ($i : 0; $i < 20; $i +: 1){
		if (($i % 5) = 0) {
			$j *: -1;
		}
		$k +: $j;
		$b1 : new Brick (0,0,0, $vertices, $i, $k);
		$retArray[$i] : $b1;
	}
	return $retArray;
}

function $main : ()
{
	Array Brick $brickArray;
	int $i; int $total;
	$total : 20;

	$brickArray: $retBrickArray();

	$printstring("printing returned array of bricks");
	for ($i : 0; $i < $total; $i +: 1) {
		$b : $brickArray[$i];
		$printstring("Printing Block: ");
		$printint($b.$colorR);
		$printint($b.$colorG);
		$printint($b.$colorB);
		$printint($b.$vertices[0]);
		$printint($b.$vertices[1]);
		$printint($b.$vertices[2]);
		$printint($b.$vertices[3]);
		$printint($b.$x);
		$printint($b.$y);
	}

}
```

Test-player.rc

This test case demonstrate the language support of an array of players:

```
function $retPlayerArray : (int $total)
{
	Array Player $retArray;
	Array int $vertices;
	Player $p1;
	int $i;

	$vertices : new Array int;
	$vertices[0] : 0;
	$vertices[1] : 0;
	$vertices[2] : 20;
	$vertices[3] : 20;
```

```
	$retArray : new Array Player;


	for ($i : 0; $i < $total; $i +: 1) {
		$p1 : new Player (255,255,255, $vertices, 0);
		$retArray[$i] : $p1;
	}

	return $retArray;
}

function $main : ()
{
	Array Player $playerArray;
	Player $p;
	int $i;
	int $total;

	$total : 30;

	$playerArray: $retPlayerArray($total);

	$printstring("printing returned array of bricks");
	for ($i : 0; $i < $total; $i +: 1) {
		$p : $playerArray[$i];
		$printstring("");
		$printint($i);
		$printint($p.$colorR);
		$printint($p.$colorG);
		$printint($p.$colorB);
		$printint($p.$y);
	}

}
```

Appendix 5 Source Code:
Scanner.mll

```
{ open Parser }

rule token = parse
[' ' '\t' '\r' '\n'] { token lexbuf }        (* Whitespace *)
| "/*" { multicomment lexbuf }                    (* Double
Comments *)
| "//" { singlecomment lexbuf }              (* Single Comments
*)
| '(' { LPAREN } | ')' { RPAREN }            (* Punctuation *)
| '{' { LBRACE } | '}' { RBRACE }
| '[' { LBRACK } | ']' { RBRACK }
| ';' { SEMI } | ',' { COMMA } | '.' { REF }
| "+:" { SHORTADD } | "-:" { SHORTMINUS }    (* Arithmetic *)
| "*:" { SHORTTIMES } | "/:" { SHORTDIVIDE }
| '+' { PLUS } | '-' { MINUS }
```

```
| '*' { TIMES } | '/' { DIVIDE }
| ':' { ASSIGN } | '=' { EQ }
| '%' { MOD }
| "!=" { NEQ } | '<' { LT }                          (* Comparison
*)
| "<=" { LEQ } | '>' { GT }
| ">=" { GEQ }
| "&&" { AND } | "||" { OR } | '!' { NOT }
| "if" { IF } | "else" { ELSE }                      (* Keywords &
types *)
| "for" { FOR } | "while" { WHILE }
| "return" { RETURN }
| "void" { TYPE("void") }
| "int" { TYPE("int") }
| "string" { TYPE("string") }
| "Array" { ARRAY }
| "Map" { MAP }
| "Player" { PLAYER }
| "Brick" { BRICK }
| "function" { FUNC }
| "new" { NEW }
| "true" { LITERALINT(1) } | "false" { LITERALINT(0) }
     (* +/- integers *)
| ('-')?['0'-'9']+ as lxm { LITERALINT(int_of_string lxm) }
     (* Literal strings *)
| '"'([^'"'] | '\\'"')*'"' as str { LITERALSTRING(String.sub
str 1 ((String.length str) - 2 )) }
     (* Identifiers *)
| '$'['a'-'z' 'A'-'Z']+['a'-'z' 'A'-'Z' '0'-'9' '_']* as lxm
{ ID(lxm) }
| eof { EOF } (* End-of-file *)
| _ as charac { raise (Failure("illegal character " ^
Char.escaped charac)) }

and multicomment = parse
"*/" { token lexbuf } (* End-of-comment *)
| eof { raise ( Failure("eof reached before multicomment
completion")) }
| _ { multicomment lexbuf } (* Eat everything else *)

and singlecomment = parse
'\n' { token lexbuf } (* End-of-comment *)
| _ { singlecomment lexbuf } (* Eat everything else *)
```

Ast.ml

```
type op = Add | Sub | Mult | Div | Mod | Equal | Neq | Less |
Leq | Greater | Geq | And | Or

type expr =
    LiteralInt of int                   (* Integers *)
  | LiteralString of string             (* Strings *)
  | Id of string                        (* Reference a variable
*)
  | Brick of expr * expr * expr * expr * expr * expr (*
Construct Brick: Brick(r, g, b, array of points, x, y) *)
  | Player of expr * expr * expr * expr * expr       (*
Construct Player: Player(r, g, b, array of points, y) *)
  | Array of string                     (* Locate the array and
initialize it by inserting a variable type identifier
                                           e.g. 6 for int array,
7 for string array *)
  | Map of expr * expr * string         (* Construct Map:
Map(height, width, generator function) *)
  | AAccess of string * expr            (* Array access:
AAccess(arrayid, index) *)
  | AAssign of string * expr * expr     (* Assign value to index
of array: AAssign(arrayid, index, value) *)
  | Binop of expr * op * expr           (* Binary operations:
Binop(value, operator, value) *)
  | Not of expr                         (* Boolean negation *)
  | Assign of string * expr             (* Assign value to
variable *)
  | Call of string * expr list          (* Call functions *)
  | Noexpr

type stmt =
    Block of stmt list                  (* Block of statements *)
  | Expr of expr                        (* Expressions *)
  | Return of expr                      (* Return expression *)
  | If of expr * stmt * stmt            (* If statements *)
  | For of expr * expr * expr * stmt    (* For loops *)
  | While of expr * stmt                (* While loops *)

type var_decl = {
    vartype : string;                   (* Variable type *)
    varname : string;                   (* Variable name *)
}

type func_decl = {
    fname : string;                     (* Function name *)
    formals : var_decl list;            (* Function parameters *)
```

```
    locals : var_decl list;             (* Function local
variables *)
    body : stmt list;                   (* Function body
statements *)
}

type program = var_decl list * func_decl list
```

Parsery.mly:

```
%{ open Ast %}
%token SEMI LPAREN RPAREN LBRACE RBRACE LBRACK RBRACK COMMA
%token PLUS MINUS TIMES DIVIDE ASSIGN
%token SHORTADD SHORTMINUS SHORTTIMES SHORTDIVIDE MOD REF
%token EQ NEQ LT LEQ GT GEQ
%token RETURN IF ELSE FOR WHILE INT
%token AND OR NOT
%token NEW FUNC ARRAY BRICK MAP PLAYER
%token <string> TYPE
%token <int> LITERALINT
%token <string> LITERALSTRING
%token <string> ID
%token EOF

/* Define associativity of tokens */
%nonassoc NOELSE
%nonassoc ELSE
%right ASSIGN
%left SHORTADD SHORTMINUS SHORTTIMES SHORTDIVIDE
%left AND OR
%left NOT
%left EQ NEQ
%left LT GT LEQ GEQ
%left PLUS MINUS
%left TIMES DIVIDE MOD
%left REF INVOKE

/* Enters at 'program' */
%start program
%type <Ast.program> program

%%

/* Program type defined in the Ast is of the form:
   var_decl list * func_decl list
   So if you see a vdecl, add to var_decl list
   If you see a fdecl, add to the func_decl list */
program:
   /* nothing */ { [], [] }
 | program vdecl { ($2 :: fst $1), snd $1 }
 | program fdecl { fst $1, ($2 :: snd $1) }

/* Type declaration must be made separately from initialization
*/
types:
   TYPE         { $1 }
```

```
 | BRICK        { "Brick" }
 | PLAYER       { "Player" }
 | MAP          { "Map" }
 | ARRAY TYPE   { "Array" ^ $2 }
 | ARRAY BRICK  { "ArrayBrick" }
 | ARRAY PLAYER { "ArrayPlayer" }
 | ARRAY MAP    { "ArrayMap" }

/* Handle functions */
fdecl:
   FUNC ID ASSIGN LPAREN formals_opt RPAREN LBRACE vdecl_list
stmt_list RBRACE
     { { fname = $2;
         formals = $5;
         locals = List.rev $8;
         body = List.rev $9
     } }

formals_opt:
    /* nothing */ { [] }
  | formal_list   { List.rev $1 }

formal_list:
    formal_decl                   { [$1] }
  | formal_list COMMA formal_decl { $3 :: $1 }

formal_decl:
    types ID         { { vartype= $1; varname= $2; } }

/* Handle variable declarations */
vdecl_list:
    /* nothing */    { [] }
  | vdecl_list vdecl { $2 :: $1 }

vdecl:
    types ID SEMI    { { vartype= $1; varname= $2; } }

/* Handle statements */
stmt_list:
    /* nothing */    { [] }
  | stmt_list stmt   { $2 :: $1 }

stmt:
    expr SEMI                     { Expr($1) }
  | RETURN expr SEMI              { Return($2) }
  | LBRACE stmt_list RBRACE       { Block(List.rev $2) }
  | IF LPAREN expr RPAREN stmt %prec NOELSE
                                  { If($3, $5, Block([])) }
```

```
  | IF LPAREN expr RPAREN stmt ELSE stmt
                                  { If($3, $5, $7) }
  | FOR LPAREN expr_opt SEMI expr_opt SEMI expr_opt RPAREN stmt
                                  { For($3, $5, $7, $9) }
  | WHILE LPAREN expr RPAREN stmt { While($3, $5) }

/* Handle expressions */
expr_opt:
    /* nothing */ { Noexpr }
  | expr          { $1 }

expr:
    LITERALINT            { LiteralInt($1) }
  | LITERALSTRING         { LiteralString($1) }
  | expr PLUS   expr      { Binop($1, Add,   $3) }
  | expr MINUS  expr      { Binop($1, Sub,   $3) }
  | expr TIMES  expr      { Binop($1, Mult,  $3) }
  | expr DIVIDE expr      { Binop($1, Div,   $3) }
  | expr MOD    expr      { Binop($1, Mod,   $3) }
  | expr EQ     expr      { Binop($1, Equal, $3) }
  | expr NEQ    expr      { Binop($1, Neq,   $3) }
  | expr LT     expr      { Binop($1, Less,  $3) }
  | expr LEQ    expr      { Binop($1, Leq,   $3) }
  | expr GT     expr      { Binop($1, Greater, $3) }
  | expr GEQ    expr      { Binop($1, Geq,     $3) }
  | ID SHORTADD expr      { Assign($1, Binop(Id($1), Add,
$3)) }
  | ID SHORTMINUS expr    { Assign($1, Binop(Id($1), Sub,
$3)) }
  | ID SHORTTIMES expr    { Assign($1, Binop(Id($1), Mult,
$3)) }
  | ID SHORTDIVIDE expr   { Assign($1, Binop(Id($1), Div,
$3)) }
  | expr AND expr         { Binop($1, And, $3) }
  | expr OR  expr         { Binop($1, Or,  $3) }
  | NOT expr              { Not($2) }
  | NEW BRICK LPAREN expr COMMA expr COMMA expr COMMA expr COMMA
expr COMMA expr RPAREN
                          /* r, g, b, varray, x, y */
                          { Brick($4, $6, $8, $10, $12, $14) }
  | NEW MAP LPAREN expr COMMA expr COMMA ID RPAREN
                          /* width, height, brick generating
function */
                          { Map($4, $6, $8) }
  | NEW PLAYER LPAREN expr COMMA expr COMMA expr COMMA expr
COMMA expr RPAREN
                          /* r, g, b, varray, y */
                          { Player($4, $6, $8, $10, $12) }
```

```
  | NEW ARRAY TYPE        { Array($3) }
  | NEW ARRAY BRICK       { Array("Brick") }
  | NEW ARRAY PLAYER      { Array("Player") }
  | NEW ARRAY MAP         { Array("Map") }
  | ID                                      { Id($1) }
  | ID REF ID                               { Id($1 ^ "." ^
$3) }
  | ID ASSIGN expr                          { Assign($1, $3) }
  | ID REF ID ASSIGN expr                   { Assign(($1 ^ "."
^ $3), $5) }
  | ID LBRACK expr RBRACK                   { AAccess($1,
$3) } /* Handle arrays */
  | ID LBRACK expr RBRACK ASSIGN expr       { AAssign($1, $3,
$6) }
  | ID REF ID LBRACK expr RBRACK            { AAccess(($1 ^
"." ^ $3), $5) }
  | ID REF ID LBRACK expr RBRACK ASSIGN expr  { AAssign(($1 ^
"." ^ $3), $5, $8) }
  | ID LPAREN actuals_opt RPAREN            { Call($1, $3) }
  | LPAREN expr RPAREN                      { $2 }

/* Handle actual values passed to functions */
actuals_opt:
    /* nothing */ { [] }
  | actuals_list  { List.rev $1 }

actuals_list:
    expr                    { [$1] }
  | actuals_list COMMA expr { $3 :: $1 }
```

ByteCode.ml:

```
module StringMap = Map.Make(String)

type bstmt =
    Litint of int          (* Push an int literal *)
  | Litstr of string       (* Push a string literal *)
  | Drp                    (* Discard a value *)
  | Bin of Ast.op          (* Perform arithmetic on two values at
the top of the stack *)
  | Lod of int             (* Fetch global variable *)
  | Str of int             (* Store global variable *)
  | Loda                   (* Load value from global array.
                              Expects array index and array
address to be on the top of the stack *)
  | Stra                   (* Stores value to global array *)
  | Lfp of int             (* Load frame pointer relative *)
  | Sfp of int             (* Store frame pointer relative *)
  | Lfpa                   (* This is the start index of this
array variable. Index is evaluated and
                              put on top of stack in an int
structure. *)
  | Sfpa                   (* Stores frame pointer of array *)
  | Jsr of int             (* Call function by absolute address
*)
  | Ent of int             (* Push FP, FP -> SP, SP += i *)
  | Rts of int             (* Restore FP, SP, consume formals,
push result *)
  | Beq of int             (* Branch relative if top-of-stack is
zero *)
  | Bne of int             (* Branch relative if top-of-stack is
non-zero *)
  | Bra of int             (* Branch relative *)
  | Make of int             (* Shift stack pointer by 1 for Player,
Map, Brick; Adds vartype_id to first space in arrays *)
  | Init of int * int * int (* Puts vartype_id into address of
variable; used for type checking *)
  | Litf of int            (* Knows to load a function address
and offset it if necessary *)
  | ProcessBlocks
  | Hlt                    (* Terminate *)
  | Nt                     (* Negate 1 or 0 on top of stack *)

type prog = {
    globals_size : int;    (* Number of global variables *)
    text : bstmt array;    (* Code for all the functions *)
  }
```

```
let string_of_stmt = function
    Litint(i) -> "Litint " ^ string_of_int i
  | Litstr(i) -> "Litstr " ^ i
  | Litf(i) -> "Litf " ^ string_of_int i
  | Drp -> "Drp"
  | Bin(Ast.Add) -> "Bin Add"
  | Bin(Ast.Sub) -> "Bin Sub"
  | Bin(Ast.Mult) -> "Bin Mul"
  | Bin(Ast.Div) -> "Bin Div"
  | Bin(Ast.Mod) -> "Bin Mod"
  | Bin(Ast.Equal) -> "Bin Eql"
  | Bin(Ast.Neq) -> "Bin Neq"
  | Bin(Ast.Less) -> "Bin Lt"
  | Bin(Ast.Leq) -> "Bin Leq"
  | Bin(Ast.Greater) -> "Bin Gt"
  | Bin(Ast.Geq) -> "Bin Geq"
  | Bin(Ast.And) -> "Bin And"
  | Bin(Ast.Or) -> "Bin Or"
  | Lod(i) -> "Lod " ^ string_of_int i
  | Str(i) -> "Str " ^ string_of_int i
  | Lfp(i) -> "Lfp " ^ string_of_int i
  | Sfp(i) -> "Sfp " ^ string_of_int i
  | Jsr(i) -> "Jsr " ^ string_of_int i
  | Ent(i) -> "Ent " ^ string_of_int i
  | Rts(i) -> "Rts " ^ string_of_int i
  | Bne(i) -> "Bne " ^ string_of_int i
  | Beq(i) -> "Beq " ^ string_of_int i
  | Bra(i) -> "Bra " ^ string_of_int i
  | Make(i)   -> "Make " ^ string_of_int i
  | Init(i, j, k) -> "Init " ^ (string_of_int i) ^ " " ^
(string_of_int j) ^ " " ^ (string_of_int k)
  | Lfpa -> "Lfpa"
  | Sfpa -> "Sfpa"
  | Loda -> "Loda"
  | Stra -> "Stra"
  | ProcessBlocks -> "ProcessBlocks"
  | Nt -> "Not"
  | Hlt    -> "Hlt"

let string_of_prog p =
  string_of_int p.globals_size ^ " global variables\n" ^
  let funca = Array.mapi
      (fun i s -> string_of_int i ^ " " ^ string_of_stmt s)
p.text
  in String.concat "\n" (Array.to_list funca)
```

Compile.ml:

```
open Ast
open Bytecode

let array_def_size = 100

(* Symbol table: Information about all the names in scope *)
type env = {
    function_index : int StringMap.t; (* Index for each function
*)
    global_index   : int StringMap.t; (* "Address" for global
variables *)
    local_index    : int StringMap.t; (* FP offset for args,
locals *)
  }

(*
  Variable type map:
  int           : 1
  string        : 2
  Brick         : 3
  Player        : 4
  Map           : 5
  Arrayint      : 6
  Arraystring   : 7
  ArrayBrick    : 8
  ArrayPlayer   : 9
  ArrayMap      : 10
  function      : 11
*)

let string_split s =
  let rec f str lst =
    try
      if (String.length str) = 0 then
        lst
      else
        let space_index = (String.index str ' ')
        and slength = (String.length str) in
        f (String.sub str (space_index + 1) (slength -
space_index - 1))
          (if (space_index = 0) then lst else (String.sub str 0
space_index :: lst))
    with Not_found -> str :: lst
  in f s [];;
```

```
(* Given a list of variable declarations, return a list of
tuples of the form:
    (space in memory, variable name) *)
(* val enum : int -> 'a list -> (int * 'a) list *)
let rec enum stride n = function
    [] -> []
  | hd::tl ->
    if stride > 0 then
      match hd.vartype with
        "int" ->    (n + 1, hd.varname) :: enum stride (n+stride
* 2) tl
      | "string" -> (n + 39, hd.varname) :: enum stride
(n+stride * 40) tl
      | "Brick" ->
        (n + 1, hd.varname ^ ".$y" ) ::
        (n + 3, hd.varname ^ ".$x" ) ::
        (n + 204, hd.varname ^ ".$vertices") ::
        (n + 206, hd.varname ^ ".$colorB") ::
        (n + 208, hd.varname ^ ".$colorG") ::
        (n + 210, hd.varname ^ ".$colorR") ::
        (n + 211, hd.varname) :: enum stride (n+stride * 212) tl
          (* Brick size : 3 * 2 int (color), 1 * 2int for vertex
array, 2 * 2 for x and y, 1 int for type (3) = 13 *)
      | "Player" ->
        (n + 1,     hd.varname ^ ".$y") ::
        (n + 202, hd.varname ^ ".$vertices") ::
        (n + 204, hd.varname ^ ".$colorB") ::
        (n + 206, hd.varname ^ ".$colorG") ::
        (n + 208, hd.varname ^ ".$colorR") ::
        (n + 209, hd.varname) :: enum stride (n+stride * 210) tl
          (* Player size :  3 * 2 int (color), 1 * 2 int for
vertex array, 1 * 2 int (y), 1 for type (4) = 11 *)
      | "Map" ->
        (n + 1,     hd.varname ^ ".$generator") ::
        (n + 3, hd.varname ^ ".$height") ::
        (n + 5, hd.varname ^ ".$width") ::
        (n + 6, hd.varname) :: enum stride (n+stride * 7) tl
          (* Map size : 1 * 2 int for generator function, 2 x 2
int (h, w), 1 for type (5) = 7 *)
      | "Arrayint" ->   (n + 2*array_def_size, hd.varname) ::
enum stride (n+stride * 2 * array_def_size + 1) tl
      | "Arraystring" -> (n + 40*array_def_size, hd.varname) ::
enum stride (n+stride * 40 * array_def_size + 1) tl
      | "ArrayBrick" ->  (n + 212*array_def_size, hd.varname) ::
enum stride (n+stride * 212 * array_def_size + 1) tl
      | "ArrayPlayer" -> (n + 210*array_def_size, hd.varname) ::
enum stride (n+stride * 212 * array_def_size + 1) tl
```

```
      | "ArrayMap" ->    (n + 7*array_def_size, hd.varname) ::
enum stride (n+stride * 7 * array_def_size + 1) tl
      | _ -> raise(Failure ("Undefined type with variable" ^
hd.varname))
    else
      match hd.vartype with
        "int" ->    (n, hd.varname) :: enum stride (n+stride *
2) tl
      | "string" -> (n, hd.varname) :: enum stride (n+stride *
40) tl
      | "Brick" ->
        (n - 210, hd.varname ^ ".$y" ) ::
        (n - 208, hd.varname ^ ".$x" ) ::
        (n - 7, hd.varname ^ ".$vertices") ::
        (n - 5, hd.varname ^ ".$colorB") ::
        (n - 3, hd.varname ^ ".$colorG") ::
        (n - 1, hd.varname ^ ".$colorR") ::
        (n, hd.varname) :: enum stride (n+stride * 212) tl
      | "Player" ->
        (n - 208, hd.varname ^ ".$y" ) ::
        (n - 7, hd.varname ^ ".$vertices") ::
        (n - 5, hd.varname ^ ".$colorB") ::
        (n - 3, hd.varname ^ ".$colorG") ::
        (n - 1, hd.varname ^ ".$colorR") ::
        (n, hd.varname) :: enum stride (n+stride * 210) tl
      | "Map" ->
        (n - 5, hd.varname ^ ".$generator" ) ::
        (n - 3, hd.varname ^ ".$height" ) ::
        (n - 1, hd.varname ^ ".$width") ::
        (n, hd.varname) :: enum stride (n+stride * 7) tl
      | "Arrayint" ->   (n, hd.varname) :: enum stride (n+stride
* 2 * array_def_size - 1) tl
      | "Arraystring" -> (n, hd.varname) :: enum stride
(n+stride * 40 * array_def_size - 1) tl
      | "ArrayBrick" ->  (n, hd.varname) :: enum stride
(n+stride * 212 * array_def_size - 1) tl
      | "ArrayPlayer" -> (n, hd.varname) :: enum stride
(n+stride * 210 * array_def_size - 1) tl
      | "ArrayMap" ->    (n, hd.varname) :: enum stride
(n+stride * 7 * array_def_size - 1) tl
      | _ -> raise(Failure ("Undefined type with variable" ^
hd.varname))


(* Given a list of variables, generate the byte code which will
initialize all
    the types of those variables (by loading a variable id) *)
let rec enumInitCommands stride n isLocal = function
```

```
    [] -> []
  | hd::tl ->
    if stride > 0 then
      match hd.vartype with
        "int" ->
        (Init (1, (n + 1), isLocal)) ::
        enumInitCommands stride (n+stride * 2) isLocal tl
      | "string" ->
        (Init (2, (n + 39), isLocal)) ::
        enumInitCommands stride (n+stride * 40) isLocal tl
      | "Brick" ->
        (Init (1, (n + 1), isLocal)) :: (* hd.varname ^ ".$y" *)
        (Init (1, (n + 3), isLocal)) :: (* hd.varname ^ ".$x" *)
        (Init (6, (n + 204), isLocal)) :: (* hd.varname ^
".$vertices" *)
        (Init (1, (n + 206), isLocal)) :: (* hd.varname ^
".$colorB" *)
        (Init (1, (n + 208), isLocal)) :: (* hd.varname ^
".$colorG" *)
        (Init (1, (n + 210), isLocal)) :: (* hd.varname ^
".$colorR" *)
        (Init (3, (n + 211), isLocal)) ::
        enumInitCommands stride (n+stride * 212) isLocal tl
          (* Brick size : 3 * 2 int (color), 1 * 2int for vertex
array, 2 * 2 for x and y, 1 int for type (3) = 13 *)
      | "Player" ->
        (Init (1, (n + 1), isLocal)) :: (* hd.varname ^ ".$y" *)
        (Init (6, (n + 202), isLocal)) :: (* hd.varname ^
".$vertices" *)
        (Init (1, (n + 204), isLocal)) :: (* hd.varname ^
".$colorB" *)
        (Init (1, (n + 206), isLocal)) :: (* hd.varname ^
".$colorG" *)
        (Init (1, (n + 208), isLocal)) :: (* hd.varname ^
".$colorR" *)
        (Init (4, (n + 209), isLocal)) ::
        enumInitCommands stride (n+stride * 210) isLocal tl
          (* Player size :  3 * 2 int (color), 1 * 2 int for
vertex array, 1 * 2 int (y), 1 for type (4) = 11 *)
      | "Map" ->
        (Init (11, (n + 1), isLocal)) :: (* hd.varname ^
".$generator" *)
        (Init (1, (n + 3), isLocal)) :: (* hd.varname ^
".$height" *)
        (Init (1, (n + 5), isLocal)) :: (* hd.varname ^
".$width" *)
        (Init (5, (n + 6), isLocal)) ::
        enumInitCommands stride (n+stride * 7) isLocal tl
```

```
          (* Map size : 1 * 2 int for generator function, 2 x 2
int (h, w), 1 for type (5) = 7 *)
      | "Arrayint" ->
        (Init (6, (n + 2*array_def_size), isLocal)) ::
        enumInitCommands stride (n+stride * 2 * array_def_size +
1) isLocal tl
      | "Arraystring" ->
        (Init (7, (n + 40*array_def_size), isLocal)) ::
        enumInitCommands stride (n+stride * 40 * array_def_size
+ 1) isLocal tl
      | "ArrayBrick" ->
        (Init (8, (n + 212*array_def_size), isLocal)) ::
        enumInitCommands stride (n+stride * 212 * array_def_size
+ 1) isLocal tl
      | "ArrayPlayer" ->
        (Init (9, (n + 210*array_def_size), isLocal)) ::
        enumInitCommands stride (n+stride * 210 * array_def_size
+ 1) isLocal tl
      | "ArrayMap" ->
        (Init (10, (n + 7*array_def_size), isLocal)) ::
        enumInitCommands stride (n+stride * 7 * array_def_size +
1) isLocal tl
      | _ -> raise(Failure ("Undefined type with variable" ^
hd.varname))
    else
      match hd.vartype with
        "int" ->
        (Init (1, n, isLocal)) ::
        enumInitCommands stride (n+stride * 2) isLocal tl
      | "string" ->
        (Init (2, n, isLocal)) ::
        enumInitCommands stride (n+stride * 40) isLocal tl
      | "Brick" ->
        (Init (1, (n - 210), isLocal)) :: (* hd.varname ^ ".$y"
*)
        (Init (1, (n - 208), isLocal)) :: (* hd.varname ^ ".$x"
*)
        (Init (6, (n - 7), isLocal)) :: (* hd.varname ^
".$vertices" *)
        (Init (1, (n - 5), isLocal)) :: (* hd.varname ^
".$colorB" *)
        (Init (1, (n - 3), isLocal)) :: (* hd.varname ^
".$colorG" *)
        (Init (1, (n - 1), isLocal)) :: (* hd.varname ^
".$colorR" *)
        (Init (3, (n), isLocal)) ::
        enumInitCommands stride (n+stride * 212) isLocal tl
      | "Player" ->
```

```
        (Init (1, (n - 208), isLocal)) :: (* hd.varname ^ ".$y"
*)
        (Init (6, (n - 7), isLocal)) :: (* hd.varname ^
".$vertices" *)
        (Init (1, (n - 5), isLocal)) :: (* hd.varname ^
".$colorB" *)
        (Init (1, (n - 3), isLocal)) :: (* hd.varname ^
".$colorG" *)
        (Init (1, (n - 1), isLocal)) :: (* hd.varname ^
".$colorR" *)
        (Init (5, (n), isLocal)) ::
        enumInitCommands stride (n+stride * 210) isLocal tl
      | "Map" ->
        (Init (-1, (n - 5), isLocal)) :: (* hd.varname ^
".$generator" *)
        (Init (1, (n - 3), isLocal)) :: (* hd.varname ^
".$height" *)
        (Init (1, (n - 1), isLocal)) :: (* hd.varname ^
".$width" *)
        (Init (5, (n), isLocal)) ::
        enumInitCommands stride (n+stride * 7) isLocal tl
      | "Arrayint" ->
        (Init (6, n, isLocal)) ::
        enumInitCommands stride (n+stride * 2 * array_def_size -
1) isLocal tl
      | "Arraystring" ->
        (Init (7, n, isLocal)) ::
        enumInitCommands stride (n+stride * 40 * array_def_size
- 1) isLocal tl
      | "ArrayBrick" ->
        (Init (8, n, isLocal)) ::
        enumInitCommands stride (n+stride * 212 * array_def_size
- 1) isLocal tl
      | "ArrayPlayer" ->
        (Init (9, n, isLocal)) ::
        enumInitCommands stride (n+stride * 210 * array_def_size
- 1) isLocal tl
      | "ArrayMap" ->
        (Init (10, n, isLocal)) ::
        enumInitCommands stride (n+stride * 7 * array_def_size -
1) isLocal tl
      | _ -> raise(Failure ("Undefined type with variable" ^
hd.varname))

(* Enumerate function pointers *)
(* val enum : int -> 'a list -> (int * 'a) list *)
let rec enum_func stride n = function
    [] -> []
```

```
  | hd::tl -> (n, hd) :: enum_func stride (n+stride) tl

(* Calculate total size of a variable list *)
let total_varsize a vlist =
   List.fold_left (fun a b -> a + (match b.vartype with
                     "int" -> 2
                   | "string" -> 40
                   | "Brick" -> 212
                   | "Player" -> 210
                   | "Map" -> 7
                   | "Arrayint" -> array_def_size*2+1
                   | "Arraystring"  -> array_def_size*40+1
                   | "ArrayBrick" -> array_def_size*212+1
                   | "ArrayPlayer" -> array_def_size*210+1
                   | "ArrayMap" -> array_def_size*7+1
                   |  _ -> raise(Failure("Error in
total_varsize"))
                  )) 0 vlist

(* Given a list of tuples of, create a StringMap for easier look
up *)
(* val string_map_pairs StringMap 'a -> (int * 'a) list ->
StringMap 'a *)
let string_map_pairs map pairs =
  List.fold_left (fun m (i, n) -> StringMap.add n i m) map pairs

(** Translate a program in AST form into a bytecode program.
Throw an
    exception if something is wrong, e.g., a reference to an
unknown
    variable or function *)
let translate (globals, functions) =

  (* Allocate "addresses" for each global variable *)
  let global_indexes = string_map_pairs StringMap.empty (enum 1
0 globals) in
  let globalinits = enumInitCommands 1 0 0 globals in
  (* Assign indexes to function names *)
  let built_in_functions = StringMap.add "$LoadPlayer" (-1)
StringMap.empty in
  let built_in_functions = StringMap.add "$Run" (-2)
built_in_functions in
  let built_in_functions = StringMap.add "$printint" (-3)
built_in_functions in
  let built_in_functions = StringMap.add "$printstring" (-4)
built_in_functions in
  let built_in_functions = StringMap.add "$dumpstack" (-5)
built_in_functions in
```

```
  let built_in_functions = StringMap.add "$CallGenerator" (-6)
built_in_functions in
  let built_in_functions = StringMap.add "$Push" (-7)
built_in_functions in
  let built_in_functions = StringMap.add "$GetCurrentScore" (-8)
built_in_functions in
  let built_in_functions = StringMap.add "$GenerateRandomInt" (-
9) built_in_functions in
  let built_in_functions = StringMap.add "$ArrayCount" (-10)
built_in_functions in

  let function_indexes = string_map_pairs built_in_functions
      (enum_func 1 1 (List.map (fun f -> f.fname) functions)) in

  (* Translate a function in AST form into a list of bytecode
statements *)
  let translate env fdecl =
    (* Bookkeeping: FP offsets for locals and arguments *)
    let num_formals = total_varsize 0 fdecl.formals
    and num_locals = total_varsize 0 fdecl.locals
    and local_offsets = enum 1 1 fdecl.locals
    and formal_offsets = enum (-1) (-2) fdecl.formals in
    let localinits = enumInitCommands 1 1 1 fdecl.locals
    and formalinits = enumInitCommands (-1) (-2) 1 fdecl.formals
in
    let env = { env with local_index = string_map_pairs
      StringMap.empty (local_offsets @ formal_offsets) } in

    (* Evaluate items from the AST into bytecode instructions *)
    let rec expr = function
        LiteralInt i -> [Litint i]
      | LiteralString i -> [Litstr i]
      | Id s ->
          (try [Lfp (StringMap.find s env.local_index)]
           with Not_found -> try [Lod (StringMap.find s
env.global_index)]
           with Not_found -> try [Litint (StringMap.find s
env.function_index)]
           with Not_found -> raise (Failure ("undeclared Id " ^
s)))
      | Brick (r, g, b, varray, x, y) ->
          expr y @ expr x
          @ (expr varray)
          @ expr b @ expr g @ expr r
          @ [Litint 3] @ [Make 3]
      | Player (r, g, b, varray, y) ->
          expr y @ (expr varray)
          @ expr b @ expr g @ expr r
```

```
          @ [Litint 4] @ [Make 4]
      | Map (width, height, generator) ->
        (try [Litf (StringMap.find generator function_indexes)]
         with Not_found -> raise (Failure ("undeclared function
" ^ generator)))
        @ expr height @ expr width @ [Litint 5] @ [Make 5]
      | Array (array_type) -> (* Push an empty array onto stack
with type identifier on top *)
          (match array_type with
                "int" -> [Make 6]
            |   "string" -> [Make 7]
            |   "Brick" -> [Make 8]
            |   "Player" -> [Make 9]
            |   "Map" -> [Make 10]
            | _ -> raise (Failure ("Invalid array type " ^
array_type))
          )
      | AAccess(a, i) ->
          expr i @
          (try [Litint (StringMap.find a env.local_index)] @
[Lfpa]
          with Not_found -> try [Litint (StringMap.find a
env.global_index)] @ [Loda]
          with Not_found -> raise (Failure ("AAccess: undeclared
array " ^ a)))
      | AAssign(a, i, e) ->
          expr e @ expr i @
          (try [Litint (StringMap.find a env.local_index)] @
[Sfpa]
          with Not_found -> try [Litint (StringMap.find a
env.global_index)] @ [Stra]
          with Not_found -> raise (Failure ("AAssign: undeclared
array " ^ a)))
      | Binop (e1, op, e2) -> expr e1 @ expr e2 @ [Bin op]
      | Not(e) -> expr e @ [Nt]
      | Assign (s, e) ->
           expr e @
          (try [Sfp (StringMap.find s env.local_index)]
          with Not_found -> try [Str (StringMap.find s
env.global_index)]
          with Not_found -> raise (Failure ("Assign: undeclared
variable " ^ s)))
      | Call (fname, actuals) ->
          if (fname = "$Run") then
            let actualVars = (List.concat (List.map expr
actuals)) in
            if (List.length actualVars) <> 2 then
raise(Failure("The function run expects 2 parameters.")) else
```

```
            let loadMap = [List.hd actualVars]
            and loadPlayer = [List.nth actualVars 1] in
            (expr (Call("$CallGenerator", [List.nth actuals
0]))) @ [ProcessBlocks] @ (expr (Call("$LoadPlayer", [List.nth
actuals 1]))) @ [Jsr (-2)]
          else
          if (fname = "$Push") then
            if (List.length actuals) <> 2 then
raise(Failure("Push requires exactly 2 arguments")) else
            let actualBytes = (List.map expr (List.rev actuals))
in
            (List.hd actualBytes) @ (match (List.hd (List.rev
(List.hd (List.rev actualBytes)))) with
                                                    Lod x ->
[Litint 0] @ [Litint x]
                                                  |  Lfp x ->
[Litint 1] @ [Litint x]
                                                  |  _ ->
raise(Failure("Invalid array specified for Push function.")))
            @ [Jsr (-7)] @ (let array_name = (match actuals with

hd :: tl -> (match hd with

Id(x) -> x

| _ -> raise(Failure("The first argument of $Push must be a
reference to an array.")))

| [] -> raise(Failure("Run must be applied to two arguments."))

      ) in
                                                   (try
[Litint (StringMap.find array_name env.local_index)] @ [Sfpa]
                                                    with
Not_found -> try[Litint (StringMap.find array_name
env.global_index)] @ [Stra]
                                                    with
Not_found -> raise (Failure ("Attempt to push onto undeclared
array " ^ array_name ^ ".")))
                                                  )
           else
           if (fname = "$GenerateRandomInt") then
              if (List.length actuals) <> 1 then
raise(Failure("You must specify a single integer argument for
the function $GenerateRandomInt.")) else
              expr (List.hd actuals) @ [Jsr (-9)]
           else
           (
```

```
            (List.concat (List.map expr (List.rev actuals))) @
            (try [Jsr (StringMap.find fname
env.function_index)]
             with Not_found -> raise (Failure ("Undefined
function: " ^ fname)))
          )
      | Noexpr -> []

    in let rec stmt = function
        Block sl     ->  List.concat (List.map stmt sl)
      | Expr e       -> expr e @ [Drp]
      | Return e     -> expr e @ [Rts num_formals]
      | If (p, t, f) -> let t' = stmt t and f' = stmt f in
                        (expr p @ [Beq(2 + List.length t')] @
                        t' @ [Bra(1 + List.length f')] @ f')
      | For (e1, e2, e3, b) -> stmt (Block([Expr(e1); While(e2,
Block([b; Expr(e3)]))]))
      | While (e, b) -> let b' = stmt b and e' = expr e in
                        [Bra (1+ List.length b')] @ b' @ e' @
                        [Bne (-(List.length b' + List.length
e'))]

    in [Ent num_locals] @           (* Entry: allocate space for
locals *)
       formalinits @ localinits @
       stmt (Block fdecl.body) @    (* Body *)
       [Litint 0; Rts num_formals]  (* Default = return 0 *)

    in let env = { function_index = function_indexes;
                   global_index = global_indexes;
                   local_index = StringMap.empty } in

  (* Code executed to start the program: Jsr main; halt *)
    let entry_function =
        try globalinits @ [Jsr (StringMap.find "$main"
function_indexes); Hlt]
        with Not_found -> raise (Failure ("no \"$main\"
function"))
    in

  (* Compile the functions *)
  let func_bodies = entry_function :: List.map (translate env)
functions in

  (* Calculate function entry points by adding their lengths *)
  let (fun_offset_list, _) = List.fold_left
      (fun (l,i) f -> (i :: l, (i + List.length f))) ([],0)
func_bodies in
```

```
  let func_offset = Array.of_list (List.rev fun_offset_list) in

  { globals_size = total_varsize 0 globals;
    (* Concatenate the compiled functions and replace the
function
       indexes in Jsr statements with PC values *)
    text = Array.of_list (List.map (function
                                      Jsr i when i > 0 -> Jsr
func_offset.(i)
                                    | Litf i when i > 0 ->
Litint func_offset.(i)
                                    | _ as s -> s) (List.concat
func_bodies))
  }
```

Execute.ml:

```
open Ast
open Bytecode
open Thread

exception IllegalMove;;
exception End;;

(*******************************************
  Structs to help organize player, block
  and brick data.
*******************************************)

type blockType = {
  mutable block_vertices:int list;
  mutable block_color:int;
  mutable block_translate_x:int;
  mutable block_translate_y:int;
};;

type playerType = {
  mutable player_vertices:int list;

  mutable player_color:int;
  mutable player_translate_y:int;
};;

type state = {
  mutable winWidth:int;
  mutable winHeight:int;
  mutable winBgColor:int;
  mutable reset: bool;
  mutable blockData: blockType list;
  mutable gravityFlag: int;
  mutable playerData: playerType;
  mutable userscore: int;
};;

(*******************************************
  Various helper functions
*******************************************)

let rec printList = function
  []  ->  ""
  | hd::tl -> (string_of_int hd) ^ printList tl;;

let array_def_size = 100
```

```
let explode s =
  let rec f acc = function
    | -1 -> acc
    | k -> f (s.[k] :: acc) (k - 1)
  in f [] (String.length s - 1) ;;

let countArray stack globals sp = (* Count array on top of stack
*)
  let rec countItems size t index count =
      if index = 100 then count else
        let itemtype = stack.(sp-2-size*index) in
        (if itemtype = 0 then (countItems size t (index+1)
count) else
          (if itemtype = t then (countItems size t (index+1)
(count+1)) else
            raise(Failure("Encountered array item of invalid
type.")))) in
          (
            match stack.(sp-1) with
                6 -> (countItems 2 1 0 0)
            |   7 -> (countItems 40 2 0 0)
            |   8 -> (countItems 212 3 0 0)
            |   9 -> (countItems 210 4 0 0)
            |   10 -> (countItems 7 5 0 0)
            |   _ -> raise(Failure("Type error: Array is of
unknown type."))
          );;

let getNextFreeIndex stack globals sp isLocal =
  let rec countItems size count t =
      if count = 100 then count else
        (if (if (isLocal <> 1) then (globals.(sp-1-size*count))
else (stack.(sp-1-size*count))) <> t then count else
         countItems size (count+1) t) in
          (
            match (if (isLocal <> 1) then (globals.(sp)) else
(stack.(sp))) with
                6 -> (countItems 2 0 1)
            |   7 -> (countItems 40 0 2)
            |   8 -> (countItems 212 0 3)
            |   9 -> (countItems 210 0 4)
            |   10 -> (countItems 7 0 5)
            |   _ -> raise(Failure("Type error: Array is of
unknown type."))
          );;
```

```
(*****************************************
  Graphics helpers
*****************************************)
let draw_polygon vlist color =
  Graphics.set_color color;
  let x0 = (List.nth vlist 0) and y0 = (List.nth vlist 1) in
    Graphics.moveto x0 y0;
    for i = 1 to ((List.length vlist) / 2) - 1 do
      let x = (List.nth vlist (2*i)) and y = (List.nth vlist
(2*i + 1)) in Graphics.lineto x y;
    done;
    Graphics.lineto x0 y0;

  let rec buildTupleArray = function
    [] -> []
    | px::py::tl -> (px,py)::(buildTupleArray tl)
    | _ :: [] -> raise(Failure("The vertices array provided does
not contain a complete set of x,y coordinates."))
  in
  Graphics.fill_poly (Array.of_list (buildTupleArray vlist));;

(* Convert (r,g,b) into a single OCaml color value c *)
let color_from_rgb r g b =
  r*256*256 + g*256 + b;;

(*
  Relatively translate all vertex given the translation distance
ex
*)
let rec trans_allVertices_x ex = function
  [] -> []
  | px::py::tl -> (px + ex)::(py::(trans_allVertices_x ex tl))
  | _ :: [] -> raise(Failure("The vertices array provided does
not contain a complete set of x,y coordinates."));;

(*
  Relatively translate all vertex given the translation distance
ey
*)
let rec trans_allVertices_y ey = function
  [] -> []
  | px::py::tl ->(px)::((py + ey)::(trans_allVertices_y ey tl))
  | _ :: [] -> raise(Failure("The vertices array provided does
not contain a complete set of x,y coordinates."));;

(*
  Given absolute location in x of the first vertex of the
polygon,
```

```
  rigidly translate all vertex relative to this absolute
location
*)
let trans_allVertices_abs_x abx vlist =
  let distant = abx - (List.nth vlist 0) in
    let rec trans_abs_x dist = function
      [] -> []
      | px::py::tl -> (px + dist)::(py::(trans_abs_x dist tl))
      | _ :: [] -> raise(Failure("The vertices array provided
does not contain a complete set of x,y coordinates.")) in
        trans_abs_x distant vlist;;

(*
  Given absolute location in y of the first vertex of the
polygon,
  rigidly translate all vertex relative to this absolute
location
*)
let trans_allVertices_abs_y aby vlist =
  let distant = aby - (List.nth vlist 1) in
    let rec trans_abs_y dist = function
      [] -> []
      | px::py::tl -> (px)::((py + dist)::(trans_abs_y dist tl))
      | _ :: [] -> raise(Failure("The vertices array provided
does not contain a complete set of x,y coordinates.")) in
        trans_abs_y distant vlist;;

(* Given the start value and the list of vertices, compute max
or min *)
let rec find_max_y current = function
    []           -> current
    | px::py::tl -> if (py > current) then (find_max_y py tl)
else (find_max_y current tl)
    | _ :: [] -> raise(Failure("The vertices array provided does
not contain a complete set of x,y coordinates."));;

let rec find_min_y current = function
    []           -> current
    | px::py::tl -> if (py < current) then (find_min_y py tl)
else (find_min_y current tl)
    | _ :: [] -> raise(Failure("The vertices array provided does
not contain a complete set of x,y coordinates."));;

let rec find_max_x current = function
    []           -> current
    | px::py::tl -> if (px > current) then (find_max_x px tl)
else (find_max_x current tl)
```

```
    | _ :: [] -> raise(Failure("The vertices array provided does
not contain a complete set of x,y coordinates."));;

let rec find_min_x current = function
    []           -> current
    | px::py::tl -> if (px < current) then (find_min_x px tl)
else (find_min_x current tl)
    | _ :: [] -> raise(Failure("The vertices array provided does
not contain a complete set of x,y coordinates."));;

(*
  Given a list of vertex coordinates [x0, y0, x1, y1, ...] and
color,
  draw and fill the polygon.
  *)
let draw_polygon vlist color =

  Graphics.set_color color;
  let x0 = (List.nth vlist 0) and y0 = (List.nth vlist 1) in
    Graphics.moveto x0 y0;
    for i = 1 to ((List.length vlist) / 2) - 1 do
      let x = (List.nth vlist (2*i)) and y = (List.nth vlist
(2*i + 1)) in Graphics.lineto x y;
    done;
    Graphics.lineto x0 y0;

  let rec buildTupleArray = function
    [] -> []
    | px::py::tl -> (px,py)::(buildTupleArray tl)
    | _ :: [] -> raise(Failure("The vertices array provided does
not contain a complete set of x,y coordinates."))
  in
  Graphics.fill_poly (Array.of_list (buildTupleArray vlist));;

(* Draw the moving block *)
let draw_rectangle x y size color =
  Graphics.set_color color;
  Graphics.fill_rect (x) (y) size size;;

let draw_string x y str =
  Graphics.moveto x y;
  Graphics.set_text_size 30;
  Graphics.draw_string str;;

let blocks1 = [];;
let player = {player_vertices = []; player_color = 0;
player_translate_y = 0};;
let gameState = {winWidth=(-1); winHeight=(-1);
```

```
                winBgColor=color_from_rgb 200 200 200;
                blockData=blocks1;
                playerData=player;
                reset=true;
                gravityFlag=0;
                userscore=2};;

(*****************************************
  Execute the program
*****************************************)

let execute_prog prog =
  let stack = Array.make 160000 0
  and globals = Array.make prog.globals_size 0
  and random = Random.self_init ()
  in

  let rec exec fp sp pc = try match prog.text.(pc) with
    Litint i  -> (* Load int literal *)
    stack.(sp) <- i ; stack.(sp+1) <- 1; exec fp (sp+2) (pc+1)
  | Litstr str -> (* Load string literal *)
      let ascii_list = List.rev (List.map Char.code (explode
str)) in
        let length = List.length ascii_list in
          if (length > 38) then raise(Failure("The maximum
string length allowed is 38.")) else
          let diff = 38 - length in
            let rec fill_string remaining =
              if (remaining > 0) then
                (stack.(sp+diff-remaining) <- 0;
                fill_string (remaining-1))
              else exec fp (sp+40) (pc+1) in
                let rec push_elements list index =
                  if List.length list > 0 then
                    (stack.(sp+diff+index) <- (List.hd list);
                    push_elements (List.tl list) (index+1))
                  else (stack.(sp+38) <- length; stack.(sp+39)
<- 2; fill_string diff)
                in
                  push_elements ascii_list 0
  | Drp -> (* Drop value/object on top of the stack *)
    let var_type_id = stack.(sp-1) in
      (
        match var_type_id with
          1 -> exec fp (sp-2) (pc+1)
        | 2 -> exec fp (sp-40) (pc+1)
        | 3 -> exec fp (sp-212) (pc+1)
        | 4 -> exec fp (sp-210) (pc+1)
```

```
        | 5 -> exec fp (sp-7) (pc+1)
        | 6 -> exec fp (sp-array_def_size*2-1) (pc+1)
        | 7 -> exec fp (sp-array_def_size*40-1) (pc+1)
        | 8 -> exec fp (sp-array_def_size*212-1) (pc+1)
        | 9 -> exec fp (sp-array_def_size*210-1) (pc+1)
        | 10 -> exec fp (sp-array_def_size*7-1) (pc+1)
        | _ -> raise(Failure("Unmatched type in Drp. Attempt to
drop type " ^ string_of_int var_type_id)))
  | Bin op -> (* Perform the operation op on the two values on
top of the stack *)
      let op1 = stack.(sp-4)
      and op1type = stack.(sp-3)
      and op2 = stack.(sp-2)
      and op2type = stack.(sp-1) in
      if ((op1type <> op2type) || (op1type <> 1)) then
raise(Failure("Binary operations can only be done on
integers.")) else
      stack.(sp-4) <- (let boolean i = if i then 1 else 0 in
      match op with
           Add     -> op1 + op2
      | Sub     -> op1 - op2
      | Mult    -> op1 * op2
      | Div     -> op1 / op2
      | Mod     -> op1 mod op2
      | Equal   -> boolean (op1 = op2)
      | Neq     -> boolean (op1 != op2)
      | Less    -> boolean (op1 <  op2)
      | Leq     -> boolean (op1 <= op2)
      | Greater -> boolean (op1 >  op2)
      | Geq     -> boolean (op1 >= op2)
      | And     -> (if op1 == 0 || op2 == 0 then 0 else 1)
      | Or      -> (if op1 == 1 || op2 == 1 then 1 else 0));
      exec fp (sp-2) (pc+1)
  | Lod i -> (* Load a global variable *)
    let var_type_id = globals.(i)
      in
        (match var_type_id with
            1 -> (* int *)
              stack.(sp) <- globals.(i-1);
              stack.(sp+1) <- globals.(i);
              exec fp (sp+2) (pc+1)
          | 2 -> (* string *)
              for j=0 to 39 do
                stack.(sp+j) <- globals.(i-39+j)
              done;
              exec fp (sp+40) (pc+1)
          | 3 ->  (* Brick *)
              for j=0 to 211 do
```

```
              stack.(sp+j) <- globals.(i-211+j)
            done;
            exec fp (sp+212) (pc+1)
        | 4 ->  (* Player *)
            for j=0 to 209 do
              stack.(sp+j) <- globals.(i-209+j)
            done;
            exec fp (sp+210) (pc+1)
        | 5 ->  (* Map *)
            for j=0 to 6 do
              stack.(sp+j) <- globals.(i-6+j)
            done;
            exec fp (sp+7) (pc+1)
        | 6 ->  (* Arrayint *)
            for j=0 to 200 do
              stack.(sp+j) <- globals.(i-200+j)
            done;
            exec fp (sp+201) (pc+1)
        | 7 ->  (* Arraystring *)
            for j=0 to 4000 do
              stack.(sp+j) <- globals.(i-4000+j)
            done;
            exec fp (sp+4001) (pc+1)
        | 8 ->  (* ArrayBrick *)
            for j=0 to 21200 do
              stack.(sp+j) <- globals.(i-21200+j)
            done;
            exec fp (sp+21201) (pc+1)
        | 9 ->  (* ArrayPlayer *)
            for j=0 to 21000 do
              stack.(sp+j) <- globals.(i-21000+j)
            done;
            exec fp (sp+21001) (pc+1)
        | 10 ->  (* ArrayMap *)
            for j=0 to 700 do
              stack.(sp+j) <- globals.(i-700+j)
            done;
            exec fp (sp+701) (pc+1)
        | _ -> raise(Failure("Type error: Attempt to load
unknown type!"))
      )
  | Str i -> (* Store a global variable *)
    let globaltypeid = globals.(i)
    and var_type_id = stack.(sp-1) in
    if (globaltypeid <> var_type_id) then raise(Failure("Attempt
to set global variable to mismatched type.")) else
    (
      match var_type_id with
```

```
  1 -> (* int *)
    globals.(i-1) <- stack.(sp-2);
    globals.(i) <- stack.(sp-1);
    exec fp (sp) (pc+1)
| 2 -> (* string *)
    for j=0 to 39 do
      globals.(i-39+j) <- stack.(sp-40+j)
    done;
    exec fp (sp) (pc+1)
| 3 -> (* Brick *)
    for j=0 to 211 do
      globals.(i-211+j) <- stack.(sp-212+j)
    done;
    exec fp (sp) (pc+1)
| 4 -> (* Player *)
    for j=0 to 209 do
      globals.(i-209+j) <- stack.(sp-210+j)
    done;
    exec fp (sp) (pc+1)
| 5 -> (* Map *)
    for j=0 to 6 do
      globals.(i-6+j) <- stack.(sp-7+j)
    done;
    exec fp (sp) (pc+1)
| 6 -> (* Arrayint *)
    for j=0 to 200 do
      globals.(i-200+j) <- stack.(sp-201+j)
    done;
    exec fp (sp) (pc+1)
| 7 -> (* Arraystring *)
    for j=0 to 4000 do
      globals.(i-4000+j) <- stack.(sp-4001+j)
    done;
    exec fp (sp) (pc+1)
| 8 -> (* ArrayBrick *)
    for j=0 to 21200 do
      globals.(i-21200+j) <- stack.(sp-21201+j)
    done;
    exec fp (sp) (pc+1)
| 9 -> (* ArrayPlayer *)
    for j=0 to 21000 do
      globals.(i-21000+j) <- stack.(sp-21001+j)
    done;
    exec fp (sp) (pc+1)
| 10 -> (* ArrayMap *)
    for j=0 to 700 do
      globals.(i-700+j) <- stack.(sp-701+j)
    done;
```

```
          exec fp (sp) (pc+1)
      | 0 -> raise(Failure("Unable to store uninitialized
variable."))
      | _ -> raise(Failure("Type error: Unable to store variable
of unknown type."))
    )
  | Loda -> (* Load a value from a global array, first element
on stack is address of array, next element is index of array *)
    if (stack.(sp-1) <> 1) then raise(Failure("Invalid array
address.")) else
    if (stack.(sp-3) <> 1) then raise(Failure("Type error: Array
index must be an integer!")) else
    let i = stack.(sp-2) (* Address of array being accessed *)
    and elem_index = stack.(sp-4) in (* Index of array to access
*)
    let var_type_id = globals.(i) in
    let cnst_offset = 4 in
    let elem_size =
      (
        match var_type_id with
          6 -> 2  (* int *)
        | 7 -> 40 (* string *)
        | 8 -> 212 (* Brick *)
        | 9 -> 210 (* Player *)
        | 10 -> 7 (* Map *)
        | 0 -> raise(Failure("Attempt to access uninitialized
global array."))
        | _ -> raise(Failure("Type error: Attempt to access the
index of a nonarray."))
      )
    in
    (match var_type_id with
        6 -> (* Arrayint *)
          stack.(sp-4) <- globals.(i-2-elem_size*elem_index);
          stack.(sp+1-4) <- globals.(i-1-elem_size*elem_index);
          exec fp (sp-2) (pc+1)
      | 7 -> (* Arraystring *)
          for j=0 to (elem_size - 1) do
            stack.(sp+j-cnst_offset) <- globals.(i-elem_size-
elem_size*elem_index+j)
          done;
          exec fp (sp+elem_size-cnst_offset) (pc+1)
      | 8 -> (* ArrayBrick *)
          for j=0 to (elem_size - 1) do
            stack.(sp+j-cnst_offset) <- globals.(i-elem_size-
elem_size*elem_index+j)
          done;
          exec fp (sp+elem_size-cnst_offset) (pc+1)
```

```
      | 9 -> (* ArrayPlayer *)
          for j=0 to (elem_size - 1) do
            stack.(sp+j-cnst_offset) <- globals.(i-elem_size-
elem_size*elem_index+j)
          done;
          exec fp (sp+elem_size-cnst_offset) (pc+1)
      | 10 -> (* ArrayMap *)
          for j=0 to (elem_size - 1) do
            stack.(sp+j-cnst_offset) <- globals.(i-elem_size-
elem_size*elem_index+j)
          done;
          exec fp (sp+elem_size-cnst_offset) (pc+1)
      | _ -> raise(Failure("Type error: Global variable accessed
is of unknown type."))
    )
  | Stra -> (* Store a value into global array, top of stack is
array address, next is array index, then value to store *)
    if (stack.(sp-1) <> 1) then raise(Failure("Invalid array
address.")) else
    if (stack.(sp-3) <> 1) then raise(Failure("Type error: Array
index must be an integer.")) else
    let array_address = stack.(sp-2)
    and obj_id = stack.(sp-5)
    and offset = stack.(sp-4) in
    let var_type_id = globals.(array_address) in
    let elem_type = (match var_type_id with
                        6 -> 1  (* int *)
                      | 7 -> 2  (* string *)
                      | 8 -> 3  (* Brick *)
                      | 9 -> 4  (* Player *)
                      | 10 -> 5 (* Map *)
                      | _ -> raise(Failure("Type error: Global
array referenced is of unknown type."))   (* Unmatched type *)
                    )
    and elem_size = (match var_type_id with
                        6 -> 2
                      | 7 -> 40
                      | 8 -> 212
                      | 9 -> 210
                      | 10 -> 7
                      | 0 -> raise(Failure("Global array
referenced is uninitialized."))   (* Uninitialized *)
                      | _ -> raise(Failure("Type error: Global
array referenced is of unknown type."))   (* Unmatched type *)
                    )
    in
    if (obj_id <> elem_type) then raise (Failure("Attempt to set
index of array to mismatched type."))
```

```
    else
    (
      match var_type_id with
        6 -> (* Arrayint *)
          for j=1 to elem_size do
            globals.(array_address-j-elem_size*offset) <-
stack.(sp-j-4)
          done;
          exec fp (sp) (pc+1)
      | 7 -> (* Arraystring *)
          for j=1 to elem_size do
            globals.(array_address-j-elem_size*offset) <-
stack.(sp-j-4)
          done;
          exec fp (sp) (pc+1)
      | 8 -> (* ArrayBrick *)
          for j=1 to elem_size do
            globals.(array_address-j-elem_size*offset) <-
stack.(sp-j-4)
          done;
          exec fp (sp) (pc+1)
      | 9 -> (* ArrayPlayer *)
          for j=1 to elem_size do
            globals.(array_address-j-elem_size*offset) <-
stack.(sp-j-4)
          done;
          exec fp (sp) (pc+1)
      | 10 -> (* ArrayMap *)
          for j=1 to elem_size do
            globals.(array_address-j-elem_size*offset) <-
stack.(sp-j-4)
          done;
          exec fp (sp) (pc+1)
      | _ -> raise(Failure("Type error: Attempt to store array
of unknown type."))
    )
  | Lfp i -> (* Load a local variable *)
      let var_type_id = stack.(fp+i) in
      let elem_size = (match var_type_id with
                          1 -> 2
                        |  2 -> 40
                        |  3 -> 212
                        |  4 -> 210
                        |  5 -> 7
                        |  6 -> 201
                        |  7 -> 4001
                        |  8 -> 21201
                        |  9 -> 21001
```

```
                |  10 -> 701) in
(
    match var_type_id with
      1 -> (* int *)
        stack.(sp) <- stack.(fp+i-1); (* value *)
        stack.(sp+1) <- stack.(fp+i); (* type *)
        exec fp (sp+2) (pc+1)
    | 2 -> (* string *)
        for j=0 to (elem_size-1) do
          stack.(sp+j) <- stack.(fp+i-(elem_size-1)+j)
        done;
        exec fp (sp+elem_size) (pc+1)
    | 3 -> (* Brick *)
        for j=0 to (elem_size-1) do
          stack.(sp+j) <- stack.(fp+i-(elem_size-1)+j)
        done;
        exec fp (sp+elem_size) (pc+1)
    | 4 -> (* Player *)
        for j=0 to (elem_size-1) do
          stack.(sp+j) <- stack.(fp+i-(elem_size-1)+j)
        done;
        exec fp (sp+elem_size) (pc+1)
    | 5 -> (* Map *)
        for j=0 to (elem_size-1) do
          stack.(sp+j) <- stack.(fp+i-(elem_size-1)+j)
        done;
        exec fp (sp+elem_size) (pc+1)
    | 6 -> (* Arrayint *)
        for j=0 to (elem_size-1) do
          stack.(sp+j) <- stack.(fp+i-(elem_size-1)+j)
        done;
        exec fp (sp+elem_size) (pc+1)
    | 7 -> (* Arrayƒstring *)
        for j=0 to (elem_size-1) do
          stack.(sp+j) <- stack.(fp+i-(elem_size-1)+j)
        done;
        exec fp (sp+elem_size) (pc+1)
    | 8 -> (* ArrayBrick *)
        for j=0 to (elem_size-1) do
          stack.(sp+j) <- stack.(fp+i-(elem_size-1)+j)
        done;
        exec fp (sp+elem_size) (pc+1)
    | 9 -> (* ArrayPlayer *)
        for j=0 to (elem_size-1) do
          stack.(sp+j) <- stack.(fp+i-(elem_size-1)+j)
        done;
        exec fp (sp+elem_size) (pc+1)
    | 10 -> (* ArrayMap *)
```

```
              for j=0 to (elem_size-1) do
                stack.(sp+j) <- stack.(fp+i-(elem_size-1)+j)
              done;
              exec fp (sp+elem_size) (pc+1)
          | 0 -> (* Uninitialized variable *)
              raise(Failure("Attempt to load uninitialized local
variable."))
          | _ -> raise(Failure("Type error: Attempt to load
variable of unknown type.")))

  | Sfp i   ->
      let localvartypeid = stack.(fp+i)
      and obj_id = stack.(sp-1) in
      let elem_size = (match obj_id with
                           1 -> 2
                         |  2 -> 40
                         |  3 -> 212
                         |  4 -> 210
                         |  5 -> 7
                         |  6 -> 201
                         |  7 -> 4001
                         |  8 -> 21201
                         |  9 -> 21001
                         | 10 -> 701
                         |  _ -> raise(Failure("Type error: Unable
to determine type of object."))) in
      if (obj_id <> localvartypeid) then raise(Failure("Attempt
to store mismatched variable type in local variable.")) else
      (
        match obj_id with
          1 -> (* int *)
            stack.(fp+i) <- stack.(sp-1);
            stack.(fp+i-1) <- stack.(sp-2);
            exec fp (sp) (pc+1)
        | 2 -> (* string *)
            for j=0 to (elem_size-1) do
              stack.(fp+i-j) <- stack.(sp-j-1)
            done;
            exec fp (sp) (pc+1)
        | 3 -> (* Brick *)
            for j=0 to (elem_size-1) do
              stack.(fp+i-j) <- stack.(sp-j-1)
            done;
            exec fp (sp) (pc+1)
        | 4 -> (* Player *)
            for j=0 to (elem_size-1) do
              stack.(fp+i-j) <- stack.(sp-j-1)
            done;
```

```
            exec fp (sp) (pc+1)
        | 5 -> (* Map *)
            for j=0 to (elem_size-1) do
              stack.(fp+i-j) <- stack.(sp-j-1)
            done;
            exec fp (sp) (pc+1)
        | 6 -> (* Arrayint *)
            for j=0 to (elem_size-1) do
              stack.(fp+i-j) <- stack.(sp-j-1)
            done;
            exec fp (sp) (pc+1)
        | 7 -> (* Arraystring *)
            for j=0 to (elem_size-1) do
              stack.(fp+i-j) <- stack.(sp-j-1)
            done;
            exec fp (sp) (pc+1)
        | 8 -> (* ArrayBrick *)
            for j=0 to (elem_size-1) do
              stack.(fp+i-j) <- stack.(sp-j-1)
            done;
            exec fp (sp) (pc+1)
        | 9 -> (* ArrayPlayer *)
            for j=0 to (elem_size-1) do
              stack.(fp+i-j) <- stack.(sp-j-1)
            done;
            exec fp (sp) (pc+1)
        | 10 -> (* ArrayMap *)
            for j=0 to (elem_size-1) do
              stack.(fp+i-j) <- stack.(sp-j-1)
            done;
            exec fp (sp) (pc+1)
        | _ -> raise(Failure("Type error: Unmatched type
error!"))
      )
  | Lfpa -> (* Load index of local array, based on next integer
on stack *)
    if (stack.(sp-1) <> 1) then raise(Failure("Invalid array
address.")) else
    if (stack.(sp-3) <> 1) then raise(Failure("Type error: Array
index must be an integer.")) else
    let i = stack.(sp-2) in (* array address *)
    let cnst_offset = 4 in
    let obj_id = stack.(fp+i)
    and loffset = stack.(sp-4) in (* array index *)
    (
      match obj_id with
        6 -> (* Arrayint *)
```

```
          if stack.(fp+i-1-loffset*2) = 0 then
raise(Failure("Attempt to load from array at an uninitialized
index.")) else
          (stack.(sp-4) <- stack.(fp+i-2-loffset*2); (* value *)
          stack.(sp+1-4) <- stack.(fp+i-1-loffset*2); (* type *)
          exec fp (sp-2) (pc+1))
      | 7 -> (* Arraystring *)
          if stack.(fp+i-1-loffset*40) = 0 then
raise(Failure("Attempt to load from array at an uninitialized
index.")) else
          (for j=0 to 39 do
            stack.(sp+j-cnst_offset) <- stack.(fp+i-40+j-
loffset*40)
          done;
          exec fp (sp+40-cnst_offset) (pc+1))
      | 8 -> (* ArrayBrick *)
          if stack.(fp+i-1-loffset*212) = 0 then
raise(Failure("Attempt to load from array at an uninitialized
index.")) else
          (for j=0 to 211 do
            stack.(sp+j-cnst_offset) <- stack.(fp+i-212+j-
loffset*212)
          done;
          exec fp (sp+212-cnst_offset) (pc+1))
      | 9 -> (* ArrayPlayer *)
          if stack.(fp+i-1-loffset*210) = 0 then
raise(Failure("Attempt to load from array at an uninitialized
index.")) else
          (for j=0 to 209 do
            stack.(sp+j-cnst_offset) <- stack.(fp+i-210+j-
loffset*210);
          done;
          exec fp (sp+210-cnst_offset) (pc+1))
      | 10 -> (* ArrayMap *)
          if stack.(fp+i-1-loffset*7) = 0 then
raise(Failure("Attempt to load from array at an uninitialized
index.")) else
          (for j=0 to 6 do
            stack.(sp+j-cnst_offset) <- stack.(fp+i-7+j-
loffset*7)
          done;
          exec fp (sp+7-cnst_offset) (pc+1))
      | 0 -> (* Uninitialized array *)
          raise(Failure("Attempt to access index of
uninitialized array."))
      | _ -> raise(Failure("Type error: Attempt to access index
of array of unknown type."))
  )
```

```
  | Sfpa -> (* Store into index of array the next item on stack
after index *)
    if (stack.(sp-1) <> 1) then raise(Failure("Invalid array
address.")) else
    if (stack.(sp-3) <> 1) then raise(Failure("Type error: Array
index must be an integer.")) else
    let i = stack.(sp-2) in (* array address *)
    let obj_id = stack.(sp-5)
    and loffset = stack.(sp-4)
    and array_type_id = stack.(fp+i) in
    if (obj_id <> (match array_type_id with
                     6 -> 1
                   | 7 -> 2
                   | 8 -> 3
                   | 9 -> 4
                   | 10 -> 5
                   | 0 -> raise (Failure("Attempt to store
value into uninitialized array."))
                   | _ -> raise (Failure("Type error: Attempt
to access array of unknown type."))))
      then raise(Failure("Type mismatch: Attempt to store value
of mismatched type into local array."))
    else
    (
      match array_type_id with
        6 -> (* Arrayint *)
           stack.(fp+i-1-2*loffset) <- stack.(sp-5);
           stack.(fp+i-2-2*loffset) <- stack.(sp-6);
           exec fp (sp) (pc+1)
      | 7 -> (* Arraystring *)
          for j=1 to 40 do
            stack.(fp+i-j-40*loffset) <- stack.(sp-j-4)
          done;
          exec fp (sp) (pc+1)
      | 8 -> (* ArrayBrick *)
          for j=1 to 212 do
            stack.(fp+i-j-212*loffset) <- stack.(sp-j-4)
          done;
          exec fp (sp) (pc+1)
      | 9 -> (* ArrayPlayer *)
          for j=1 to 210 do
            stack.(fp+i-j-210*loffset) <- stack.(sp-j-4)
          done;
          exec fp (sp) (pc+1)
      | 10 -> (* ArrayMap *)
          for j=1 to 7 do
            stack.(fp+i-j-7*loffset) <- stack.(sp-j-4)
          done;
```

```
          exec fp (sp) (pc+1)
      | _ -> raise(Failure("Type error: Attempt to store value
into array of unknown type."))
    )
  | Jsr(-1) -> (* DrawPlayer *)
      let scope = -1
      and addr = (sp-9)
      and color = color_from_rgb stack.(sp-3) stack.(sp-5)
stack.(sp-7)
      and trans_y = stack.(sp-210)
      in
        let rec make_coord_list n =
          if (scope = -1) then (*LOCAL*)
            (match stack.(fp+n) with
                0 -> []
              | 1 -> stack.(fp+n-1) :: make_coord_list (n-2)
              | _ -> raise(Failure("cant resolve " ^
string_of_int stack.(fp+n))))
          else if (scope = 1) then (*GLOBAL*)
            (match globals.(n) with
              0 -> []
            | 1 -> globals.(n-1) :: make_coord_list (n-2)
            | _ -> raise(Failure("cant resolve " ^ string_of_int
globals.(n))))
          else [] in

        let player = {player_vertices= make_coord_list (addr-
1);player_color = color; player_translate_y = trans_y} in
        gameState.playerData <- player;
        exec fp sp (pc+1)
  | Jsr(-2) -> (* Run *)

(*************************************************************
**********************************************)
(*************************************************************
**********************************************)
(*************************************************************
**********************************************)

(* s is state *)
let t_init s () =
  Graphics.open_graph (" " ^ (string_of_int s.winWidth) ^ "x" ^
                               (string_of_int s.winHeight));
  Graphics.set_color s.winBgColor;
  Graphics.fill_rect 0 0 s.winWidth s.winHeight;
  (*Graphics.set_color s.player_color;*)
  s.playerData.player_vertices <- (trans_allVertices_y
s.playerData.player_translate_y s.playerData.player_vertices);
```

```
  draw_polygon s.playerData.player_vertices
s.playerData.player_color;


  List.iter (fun block -> (block.block_vertices <-
(trans_allVertices_y block.block_translate_y
block.block_vertices))) s.blockData;

  List.iter (fun block -> (draw_polygon block.block_vertices
                                          block.block_color))
s.blockData;
  (* Debugging for graphics
  print_endline(string_of_int 1);
  draw_rectangle s.block1_x s.block1_y s.block1_size
s.block1_color;*)
in

(* s is state *)
let t_end s () =
  (* Debugging for graphics
  print_endline(string_of_int 2);*)
  Graphics.close_graph ();
  Graphics.set_color s.winBgColor;
in

(* c is keyboad input (char) *)
let t_key s c =
  (* Debugging for graphics
  print_endline(string_of_int 3);
  draw_player s.player_x s.player_y s.player_size
s.player_color;*)

  let max_y = find_max_y 0 s.playerData.player_vertices
  and min_y = find_min_y s.winHeight
s.playerData.player_vertices in
  (*let objectheight = (max_y - (List.nth
s.playerData.player_vertices 1)) in*)
  let objectheight = (max_y - min_y) in

    (match c with
      ' '   -> if max_y < s.winHeight then
               (
                  if (s.gravityFlag < 2) then
                      s.gravityFlag <- 2;
                  s.playerData.player_vertices <-
(trans_allVertices_y s.gravityFlag
s.playerData.player_vertices);
                  s.gravityFlag <- (s.gravityFlag + 3);
```

```
               )
               else
                  s.playerData.player_vertices <-
                  (trans_allVertices_abs_y (s.winHeight -
objectheight) s.playerData.player_vertices)
      |'p'   -> Thread.join(Thread.create(Thread.delay)(5.0));
      | _     -> ());
in


let t_updateFrame s () =
  (* Debugging for graphics
  print_endline(string_of_int 4); *)
  Graphics.clear_graph ();
  Graphics.set_color s.winBgColor;
  Graphics.fill_rect 0 0 s.winWidth s.winHeight;

  (*
  s.block1_x <- s.block1_x - 3;
  draw_rectangle s.block1_x s.block1_y s.block1_size
s.block1_color;
  *)

    (*
    let rec trans_allVertices = function
      [] -> []
      | px::py::tl -> (px - 3)::(py::(trans_allVertices tl))
    in*)
  List.iter (fun block -> ( block.block_vertices <-
                       (trans_allVertices_x (-10)
block.block_vertices))) s.blockData;

  List.iter (fun block -> (draw_polygon block.block_vertices
                                        block.block_color))
s.blockData;

  (* Gravitify *)
  s.gravityFlag <- (s.gravityFlag - 1);
  let max_y = find_max_y 0 s.playerData.player_vertices
  and min_y = find_min_y s.winHeight
s.playerData.player_vertices in
  let objectheight = (max_y - (List.nth
s.playerData.player_vertices 1)) in
      if (max_y > s.winHeight) then
             s.playerData.player_vertices <-
                  (trans_allVertices_abs_y (s.winHeight -
objectheight) s.playerData.player_vertices)
      else
```

```
        if (min_y > 0) then
          s.playerData.player_vertices <- (trans_allVertices_y
s.gravityFlag s.playerData.player_vertices)
        else
          s.playerData.player_vertices <-
                  (trans_allVertices_abs_y 0
s.playerData.player_vertices);
  (* End Gravitify *)


  (* Wrap map *)
    List.iter (fun block ->
               let block_max_x = (find_max_x 0
block.block_vertices) in
                if (block_max_x = 0) then (
                  ( block.block_vertices <-
(trans_allVertices_abs_x (3*s.winWidth/2)
block.block_vertices)))) s.blockData;

  (* End wrap map *)

  s.userscore <- s.userscore + 1;
  draw_string 10 (s.winHeight-20) ("Score: " ^ string_of_int
gameState.userscore);

  draw_polygon s.playerData.player_vertices
s.playerData.player_color;

in


let t_except s ex = ();
in

let t_playerCollided s () =
  (* Debugging for graphics
  print_endline(string_of_int 5);*)
  (* Get blockType block and return a GPC polygon *)
  let makeGPCPolygon vlist =
   let rec makeVertexArray = function
      []           -> [||]
      | px::py::tl -> Array.append [|{Clip.x = (float_of_int
px); Clip.y = (float_of_int py)}|] (makeVertexArray tl) in
        Clip.make_gpcpolygon [|false|] [|(makeVertexArray
vlist)|] in

  let checkCollision block =
```

```
      let _result = Clip.gpcml_clippolygon
                      Clip.Intersection
                      (makeGPCPolygon
s.playerData.player_vertices)
                      (makeGPCPolygon block.block_vertices)
                    in
                      (Clip.gpcml_isOverlapped _result)
      in

        let result list = List.fold_left (fun a b -> a || b)
false list in
          let collisionList = List.map checkCollision
s.blockData in
            result collisionList;
in

(*let i = ref 0; in*)

let skel f_init f_end f_key f_updateFrame f_except
f_playerCollided =
  (* Debugging for graphics
  print_endline(string_of_int 6);*)
  f_init ();
  try
      while not (f_playerCollided ()) do
        try

          if Graphics.key_pressed () then f_key
(Graphics.read_key ());
          (*if f_playerCollided () then f_end ();*)
          Thread.join(Thread.create(Thread.delay)(1.0 /. 24.0));
          f_updateFrame ();
        with
             End -> raise End
           |  e  -> f_except e
      done
  with
      End  -> f_end ();

in
let slate () =
    skel (t_init gameState) (t_end gameState)
         (t_key gameState) (t_updateFrame gameState)
         (t_except gameState) (t_playerCollided gameState);
in

slate ();
print_endline("Game End!");
```

```
  | Jsr(-3) -> (* printint *)
      if (stack.(sp-1) <> 1) then raise(Failure("The function
$printint must take an integer value.")) else
      print_endline (string_of_int stack.(sp-2)) ; exec fp sp
(pc+1)
  | Jsr(-4) -> (* printstring *)
      let var_type_id = stack.(sp-1) in
      if var_type_id <> 2 then raise (Failure("Type error:
Unable to call printstring on nonstring."))
      else let strLen = stack.(sp-2) in
              let rec buildStr remaining str = if (remaining >
0) then
                  buildStr (remaining-1) ((Char.escaped
(char_of_int (stack.(sp-remaining-2)))) ^ str) else str in
                  print_endline (buildStr strLen "");
                  exec fp sp (pc+1)
  | Jsr(-5) -> (* dumpstack *)
      Array.iter print_endline (Array.map string_of_int stack);
  | Jsr(-6) -> (* Jump to CallGenerator function of the map on
top of stack *)
        gameState.winWidth <- stack.(sp-3);
        gameState.winHeight <- stack.(sp-5);
        stack.(sp)   <- pc + 1 ;
        let i = stack.(sp-7) in
        exec fp (sp+1) i
  | Jsr(-7) -> (* Push function to push object on top of stack
into array *)
      let array_address = stack.(sp-2) in
      let varsize = (match (stack.(sp-5)) with
                        1 -> 2
                      |  2 -> 40
                      |  3 -> 212
                      |  4 -> 210
                      |  5 -> 7
                      |  _ -> raise(Failure("Unable to push
object of unknown type onto array."))
                    ) in
        let isLocal = (stack.(sp-4)) in
        let nextIndex = (getNextFreeIndex stack globals (if
isLocal <> 1 then (array_address) else (fp + array_address))
isLocal) in
          (if nextIndex > 100 then raise(Failure("Unable to push
value onto full array.")) else
            stack.(sp-4) <- nextIndex; stack.(sp-3) <- 1; exec
fp (sp-2) (pc+1)
          )
```

```
  | Jsr(-8) -> (* GetCurrentScore function to put current score
on stack *)
      (*stack.(sp) <- gameState.userscore; stack.(sp+1) <- 1;
exec fp (sp+2) (pc+1)*)
      stack.(sp) <- 1; stack.(sp+1) <- 1; exec fp (sp+2) (pc+1)
  | Jsr(-9) -> (* GenerateRandomInt function to generate a
random integer and put it on top of stack *)
      let seedtype = stack.(sp-1)
      and seed = stack.(sp-2) in
      if (seedtype <> 1) then raise(Failure("Type error: The
function $GenerateRandomInt requires an integer parameter."))
else
      let generated = (Random.int seed) in
      stack.(sp-2) <- generated; stack.(sp-1) <- 1; exec fp (sp)
(pc+1)
  | Jsr(-10) -> (* ArrayCount function to put number of elements
in array on stack *)
        let arraycount = (countArray stack globals sp)
        and arrayType = stack.(sp-1) in
        if (arrayType < 6 || arrayType > 10) then
raise(Failure("Unable to count the elements of a nonarray
object.")) else
        (stack.(sp) <- arraycount; stack.(sp+1) <- 1; exec fp
(sp+2) (pc+1))
  | Jsr i   -> stack.(sp)   <- pc + 1       ; exec fp (sp+1) i
  | Ent i   -> stack.(sp)   <- fp           ; exec sp (sp+i+1)
(pc+1)
  | Rts i   ->
    let new_fp = stack.(fp) and new_pc = stack.(fp-1) and base =
fp-i-1 in
    ( let obj_id = stack.(sp-1) in
      match obj_id with
        1 -> (* int *)
              (stack.(base+1) <- stack.(sp-1);  (* Construct an
int on top of stack*)
              stack.(base) <- stack.(sp-2);
              exec new_fp (base+2) new_pc)

      | 2 -> (* string *)
            (for j=0 to 39 do
              stack.(base+j) <- stack.(sp-(40-j))
            done;
            exec new_fp (base+40) new_pc)

      | 3 -> (* Brick *)
            (for j=0 to 211 do
              stack.(base+j) <- stack.(sp-(212-j))
            done;
```

```
            exec new_fp (base+212) new_pc)

      | 4 -> (* Player *)
            (for j=0 to 209 do
              stack.(base+j) <- stack.(sp-(210-j))
            done;
            exec new_fp (base+210) new_pc)

      | 5 -> (* Map *)
            (for j=0 to 6 do
              stack.(base+j) <- stack.(sp-(7-j))
            done;
            exec new_fp (base+7) new_pc)

      | 6 -> (* Arrayint *)
            (for j=0 to 200 do
              stack.(base+j) <- stack.(sp-(201-j))
            done;
            exec new_fp (base+201) new_pc)

      | 7 -> (* Arraystring *)
            (for j=0 to 4000 do
              stack.(base+j) <- stack.(sp-(4001-j))
            done;
            exec new_fp (base+4001) new_pc)

      | 8 -> (* ArrayBrick *)
           (for j=0 to 21200 do
             stack.(base+j) <- stack.(sp-(21201-j))
           done;
           exec new_fp (base+21201) new_pc)

      | 9 -> (* ArrayPlayer *)
           (for j=0 to 21000 do
             stack.(base+j) <- stack.(sp-(21001-j))
           done;
           exec new_fp (base+21001) new_pc)

       | 10 ->  (* ArrayMap *)
            (for j=0 to  700 do
              stack.(base+j) <- stack.(sp-(701-j))
            done;
            exec new_fp (base+701) new_pc)

      | _ -> raise(Failure("Unmatched type in Rts: " ^
string_of_int obj_id));
      );
```

```
  | Beq i   -> exec fp (sp-1) (pc + if stack.(sp-2) =  0 then i
else 1)
  | Bne i   -> exec fp (sp-2) (pc + if stack.(sp-2) != 0 then i
else 1)
  | Bra i   -> exec fp sp (pc+i)
  | Make id   ->
    (match id with
        0   -> exec fp (sp-1) (pc+1)
      | 1   -> raise(Failure("'Make' not required for int"));
      | 2   -> raise(Failure("'Make' not required for string"));
      | 3   -> exec fp (sp-1) (pc+1)
      | 4   -> exec fp (sp-1) (pc+1)
      | 5   -> exec fp (sp-1) (pc+1)
      | 6   -> stack.(sp+200)   <- id ; exec fp (sp+201) (pc+1)
      | 7   -> stack.(sp+4000)   <- id ; exec fp (sp+4001)
(pc+1)
      | 8   -> stack.(sp+21200)   <- id ; exec fp (sp+21201)
(pc+1)
      | 9   -> stack.(sp+21000)   <- id ; exec fp (sp+21001)
(pc+1)
      | 10  -> stack.(sp+700)   <- id ; exec fp (sp+701) (pc+1)
      | _   -> raise(Failure("'Make' cannot apply to the invalid
type " ^ string_of_int id));
    )
  | Init (i, j, k) ->
      if (k <> 1) then
        (globals.(j) <- i; exec fp sp (pc+1))
      else
        (stack.(fp+j) <- i; exec fp sp (pc+1))
  | ProcessBlocks -> (*Blocks are on the top of the stack*)
      let rec addToBricks i =
        if (stack.(i-1) = 3) then
          let scope = -1
          and r = stack.(i-3)
          and g = stack.(i-5)
          and b = stack.(i-7)
          and addr = (i-9)
          and xcoord = stack.(i-210)
          and ycoord = stack.(i-212) in

          let rec make_coord_list n =
            if (scope = -1) then (*LOCAL*)
                match stack.(fp+n) with
                 1 -> (stack.(fp+n-1)+xcoord)::((stack.(fp+n-
3)+ycoord):: make_coord_list (n-4))
               |  _ -> []
            else if (scope = 1) then (*GLOBAL*)
              (match globals.(n) with
```

```
                0 -> []
              | 1 -> (globals.(n-1)+xcoord)::((globals.(n-
3)+ycoord)::make_coord_list (n-4))
              | _ -> raise(Failure("cant resolve " ^
string_of_int globals.(n))))
            else [] in
            (
              {block_vertices= make_coord_list (addr-1);
block_color=r*256*256+g*256+b; block_translate_x=xcoord;
block_translate_y=ycoord} :: addToBricks (i-212)
            )
          else []
      in
      let blocks1 = addToBricks (sp-1) in
      gameState.blockData <- blocks1;

      (* Debugging code for loading blocks
      print_endline ("Blocks : " ^ (string_of_int (List.length
(addToBricks (sp-1)))));
      print_endline (String.concat " " (List.map string_of_int
((List.hd blocks1).block_vertices)));
      *)
      exec fp sp (pc+1)
  | Nt ->
    if (stack.(sp-1) <> 1) then
      raise(Failure("Cannot apply 'Not' to non-int")) else
        if (stack.(sp-2) = 1) then stack.(sp-2) <- 0
        else if stack.(sp-2) = 0 then stack.(sp-2) <- 1
        else raise(Failure("'Not' can only apply to 1 or 0"));
      exec fp sp (pc+1)
  | Hlt     -> ()
with _ as error -> print_endline ("Execution error: " ^
(Printexc.to_string error) ^ " at PC " ^ (string_of_int pc) ^ ".
Check the bytecode output with -b option.")
  in exec 0 0 0
```

Retrocraft.ml:

```
type action = (*Ast |*) Bytecode | Compile

let _ =
  let action = if Array.length Sys.argv > 1 then
    List.assoc Sys.argv.(1) [ (*("-a", Ast);*)
                     ("-b", Bytecode);
                     ("-c", Compile) ]
  else Compile in
  let lexbuf = Lexing.from_channel stdin in
  let program = Parser.program Scanner.token lexbuf in
  match action with
    (*Ast -> let listing = Tests.string_of_program program
           in print_string listing
  |*) Bytecode -> let listing =
      Bytecode.string_of_prog (Compile.translate program)
    in print_endline listing
  | Compile -> Execute.execute_prog (Compile.translate program)
```

Makefile:

```
OBJS = ast.cmo scanner.cmo parser.cmo bytecode.cmo compile.cmo
execute.cmo
COMPILER = retrocraft.cmo
TESTS = tests.cmo runtest.cmo

CONF=-I +threads
LIBS=$(WITHGRAPHICS) $(WITHUNIX) $(WITHTHREADS) $(WITHGPC)

# Default setting of the WITH* variables. Should be changed if
your
# local libraries are not found by the compiler.
WITHGRAPHICS =graphics.cma -cclib -lgraphics -cclib -
L/usr/X11R6/lib -cclib -lX11

WITHTHREADS =threads.cma -cclib -lthreads

WITHUNIX =unix.cma -cclib -lunix

WITHGPC =camlgpc.cma -cclib -L.


default:
     @make -f MakefileGPC
     @make compiler
```

```
compiler : $(OBJS) $(COMPILER)
     ocamlc $(CONF) -o retrocraft $(LIBS) $(OBJS) $(COMPILER)

runtests : $(OBJS) $(TESTS)
     ocamlc -o runtests $(OBJS) $(TESTS)

scanner.ml : scanner.mll
     ocamllex scanner.mll

parser.ml parser.mli : parser.mly
     ocamlyacc parser.mly

%.cmo : %.ml
     ocamlc $(CONF) -c $<

%.cmi : %.mli
     ocamlc -c $<


.PHONY : clean
clean :
     rm -rf *.cmo *.cmi retrocraft parser.mli parser.ml
scanner.ml \
     *.cmo *.cmi *.out *.diff *.a
     @make clean -f MakefileGPC

clean_runtests :
     rm -rf *.cmo *.cmi runtests parser.mli parser.ml scanner.ml

# Clean test reference files
clean_r :
     find . -name *.reference | xargs /bin/rm -f

# Clean intermediate test files
clean_i :
     find . -name *.reference | xargs /bin/rm -f
     rm -rf *.c.out *.diff

# Generated by ocamldep *.ml *.mli
ast.cmo:
ast.cmx:
bytecode.cmo: ast.cmo
bytecode.cmx: ast.cmx
compile.cmo: bytecode.cmo ast.cmo
compile.cmx: bytecode.cmx ast.cmx
execute.cmo: bytecode.cmo ast.cmo
execute.cmx: bytecode.cmx ast.cmx
```

```
#interpret.cmo: ast.cmo
#interpret.cmx: ast.cmx
retrocraft.cmo: scanner.cmo parser.cmi execute.cmo compile.cmo \
    bytecode.cmo ast.cmo tests.cmo
retrocraft.cmx: scanner.cmx parser.cmx execute.cmx compile.cmx \
    bytecode.cmx ast.cmx tests.cmx
parser.cmo: ast.cmo parser.cmi
parser.cmx: ast.cmx parser.cmi
scanner.cmo: parser.cmi
scanner.cmx: parser.cmx
parser.cmi: ast.cmo
tests.cmo:
tests.cmx:
```