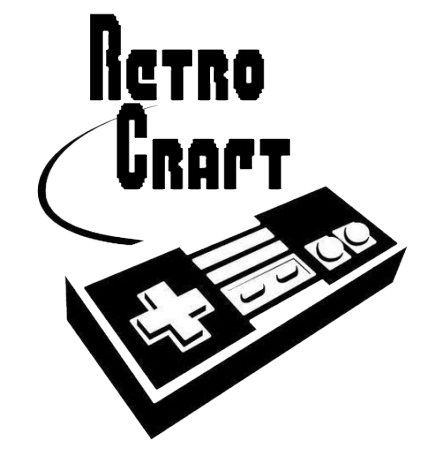
****

**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);

}



**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;

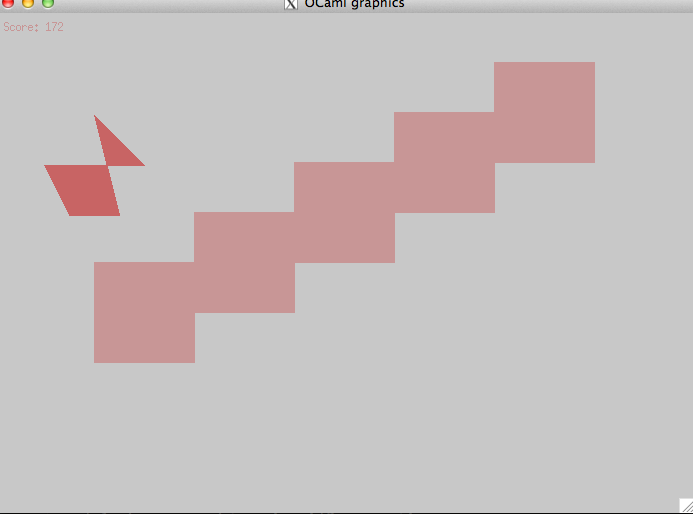
$v[9] : 100;

$p : new Player(200,100,100,$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 Brick and Player in the game. It also contains the generator function pointer that invokes function to build all blocks.  *Variable and Object Attributes*   |  |  | | --- | --- | | int $width | Width of the game screen | | int $height | Height of the game screen |   *Function Attributes*   |  |  | | --- | --- | | $generateThis | Function pointer that returns an array of blocks | |
| Brick | Fundamental building blocks of the game environment. User provides parameters: (R, G, B, $verticesArray, x, y).  *Variable and Object Attributes*   |  |  | | --- | --- | | int $colorR  int $colorG  int $colorB | User provided RGB values of the brick. | | int $x,  int $y | Translation coordinates of object | | Array int $verticesArray | pointer to an array of integers (vertices array) |   Brick objects will be translated along the map internally to simulate movement. Its movement is independent from Player. |
| Player | 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*   |  |  | | --- | --- | | int $colorR  int $colorG  int $colorB | User provided RGB values of the Player object. | | 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 ()

{

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 ()

{

Array int $retArray;

int $i;

$retArray : new Array int;

for ($i : 0; $i < 15; $i +: 1){

$retArray[$i] : $i;

}

return $retArray;

}

function $main : void ()

{

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]);

}

}

**Appendix 2:**

**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 ()

{

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 : void ()

{

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);

}

}

**Appendix 3:**

**Test-player.rc**

function $retPlayerArray : Array Player (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 : void ()

{

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 4:**

**Test-map.rc**

function $generateThis : Array Brick ()

{

Array Brick $retArray;

Array int $vertices;

Brick $b1;

int $i;

int $j;

int $k;

$vertices : new Array int;

$vertices[0] : 0;

$vertices[1] : 0;

$vertices[2] : 10;

$vertices[3] : 10;

$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 $retMapArray : Array Map (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;

}

function $main : void ()

{

Array Map $mapArray;

Map $m;

int $i; int $total;

$total : 3;

$mapArray: $retMapArray($total);

$printstring("printing returned array of maps");

for ($i : 0; $i < $total; $i +: 1) {

$m : $mapArray[$i];

$printstring("");

$printint($i);

$printint($m.$height);

$printint($m.$width);

}}

**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 }

| "\*:" { SHORTTIMES } | "/:" { SHORTDIVIDE }

| '+' { PLUS } | '-' { MINUS }

| '\*' { TIMES } | '/' { DIVIDE }

| ':' { ASSIGN } | '=' { EQ }

| '%' { MOD }

| "!=" { NEQ } | '<' { LT }

| "<=" { LEQ } | '>' { GT }

| ">=" { GEQ } | "if" { IF } (\* keywords \*)

| "&&" { AND } | "||" { OR } | '!' { NOT }

| "else" { ELSE } | "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 }

| "true" { LITERALINT(1) } | "false" { LITERALINT(0) }

| ('-')?['0'-'9']+ as lxm { LITERALINT(int\_of\_string lxm) } (\* +/- integers \*)

(\*| ['0'-'9']\*'.'['0'-'9']+ as lxm { LITERALFLOAT(float\_of\_string lxm) } (\* floats \*)\*)

(\*| ("'\\''" | '\''[^'\'''\t''\r''\n']'\'') as chr { LITERALCHAR((String.sub chr 1 ((String.length chr) - 2 )).[0]) }\*)

| '"'([^'"'] | '\\''"')\*'"' as str { LITERALSTRING(String.sub str 1 ((String.length str) - 2 )) }

| "new" { NEW }

| '$'['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

(\* TODO: Compiler will complain if we don't use "string" in Map and Brick \*)

type expr =

LiteralInt of int (\* Integers \*)

| LiteralString of string (\* Strings \*)

| Id of string (\* reference a variable \*)

| Brick of expr \* expr \* expr \* string \* expr \* expr (\* construct a Brick: Brick(r, g, b, array of points, x, y) \*)

| Player of expr \* expr \* expr \* string \* expr (\* construct Player: Player(r, g, b, array of points, y) \*)

| Array of string

| Map of expr \* expr \* string (\* construct Map: Map(height, width, generator function) \*)

| AAccess of string \* expr (\* array access: AAccess(arrayname, index) \*)

| AAssign of string \* expr \* expr (\* assign value to index of array: AAssign(arrayid, index, value) \*)

| AAccessByRef of string \* expr (\* array access: AAccess(arrayname, index) \*)

| AAssignByRef 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 \*)

| AssignToRef of string \* expr (\* assign value to variable \*)

| 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 \*)

rettype : string; (\* return type \*)

}

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

/\* Should I define LITERAL for FLOAT, STRING, etc too? \*/

%token <string> ID

%token EOF

%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

%start program

%type <Ast.program> program

%%

program:

/\* nothing \*/ { [], [] }

| program vdecl { ($2 :: fst $1), snd $1 }

| program fdecl { fst $1, ($2 :: snd $1) }

/\*

TODO: Allow vdecl\_list to mix with body?

\*/

types:

TYPE { $1 }

| BRICK { "Brick" }

| PLAYER { "Player" }

| MAP { "Map" }

| ARRAY TYPE { "Array" ^ $2 }

| ARRAY BRICK { "ArrayBrick" }

| ARRAY PLAYER { "ArrayPlayer" }

| ARRAY MAP { "ArrayMap" }

fdecl:

FUNC ID ASSIGN types LPAREN formals\_opt RPAREN LBRACE vdecl\_list stmt\_list RBRACE

{ { fname = $2;

formals = $6;

locals = List.rev $9;

body = List.rev $10;

rettype = $4 } }

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; } }

vdecl\_list:

/\* nothing \*/ { [] }

| vdecl\_list vdecl { $2 :: $1 }

vdecl:

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

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) }

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 ID 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

{ Map($4, $6, $8) }

| NEW PLAYER LPAREN expr COMMA expr COMMA expr COMMA ID COMMA expr RPAREN

{ 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 { AssignToRef(($1 ^ "." ^ $3), $5) } /\* Assignment for regular cases and the special case: reference = &obj \*/

| ID LBRACK expr RBRACK { AAccess($1, $3) }

| ID LBRACK expr RBRACK ASSIGN expr { AAssign($1, $3, $6) }

| ID REF ID LBRACK expr RBRACK { AAccessByRef(($1 ^ "." ^ $3), $5) } /\* Array w/in a struct: $brick1.vertices[0] \*/

| ID REF ID LBRACK expr RBRACK ASSIGN expr { AAssignByRef(($1 ^ "." ^ $3), $5, $8) }

| ID LPAREN actuals\_opt RPAREN { Call($1, $3) }

| LPAREN expr RPAREN { $2 }

actuals\_opt:

/\* nothing \*/ { [] }

| actuals\_list { List.rev $1 }

actuals\_list:

expr { [$1] }

| actuals\_list COMMA expr { $3 :: $1 }