ROLL

a language for designing board games

Lauren Pully (Project Manager)
Jesse Bentert (Tools & Language Guru)
John Graham (System Architect)
Daniel Wilkey (System Integrator)
Yipeng Huang (Tester & Validator)
Overview
John Graham (System Architect)

Outline of a ROLL Program
Jesse Bentert (Tools & Language Guru)

ROLL Syntax
Daniel Wilkey (System Integrator)

Compiler Architecture & Testing
Yipeng Huang (Tester & Validator)

Conclusion & Demo
Lauren Pully (Project Manager)
ROLL
A language to design and play board games
Features of ROLL

Flexible

Dynamic

Extendable
Simple

Eliminates Overhead

Code is easy to understand
Overview
John Graham (System Architect)

Outline of a ROLL Program
Jesse Bentert (Tools & Language Guru)

ROLL Syntax
Daniel Wilkey (System Integrator)

Compiler Architecture & Testing
Yipeng Huang (Tester & Validator)

Conclusion & Demo
Lauren Pully (Project Manager)
Outline of a ROLL program
Game Chutes {
Players {
   NumPieces
   StartOn
   setupPlayers() {
      NumPlayers
      Player Names
   }
}
Board {
   NumTiles
   make Tile()
   landsOn() {
      Jump
   }
   goalCheck() {
      declareWinner
   }
}
Dice {
   make Die()
   roll() {
      move
      moveReverse
   }
}
}
The Board Block

Board {
NumTiles
make Tile()
landsOn() {
Jump
}
goalCheck() {
declareWinner
}
}

Dice {
make Die()
roll() {
move
moveReverse
}
}

Game Chutes {
Players {
NumPieces
StartOn
setupPlayers() {
NumPlayers
Player Names
}
}
Board {
NumTiles
make Tile()
landsOn() {
Jump
}
goalCheck() {
declareWinner
}
}
Game Chutes {
Players {
  NumPieces
  StartOn
  setupPlayers() {
    NumPlayers
    Player Names
  }
}
Board {
  NumTiles
  make Tile()
  landsOn() {
    Jump
  }
  goalCheck() {
    declareWinner
  }
}
Dice {
  make Die()
  roll() {
    move
    moveReverse
  }
}
Overview
John Graham (System Architect)

Outline of a ROLL Program
Jesse Bentert (Tools & Language Guru)

ROLL Syntax
Daniel Wilkey (System Integrator)

Compiler Architecture & Testing
Yipeng Huang (Tester & Validator)

Conclusion & Demo
Lauren Pully (Project Manager)
GUI = 1;

Players {
    GUI = 1;
    MaxPlayers = 6;
    MinPlayers = 2;
    NumPieces = 1;
    StartOn = {0,0,0,0,0,0};
    define setupPlayers() {
        print("How many people are playing this game?");
        promptRange(NumPlayers, MinPlayers, MaxPlayers);
        for(int i : {1 ~ NumPlayers}) {
            print("What is player " | i | ", " | i | "'s name?");
            promptText(PlayerList[i-1].name);
        }
    }
}
Board {
    NumTiles = 10;

    make Tile(id: 1, next: 2, prev: 0, accessible:{5~9}, landsOn: ladder);
    make Tile(id: 3, next: 4, prev: 2, accessible:{8}, landsOn: ladder);
    make Tile(id: 6, next: 7, prev: 5, accessible:{0}, landsOn: chute);
    make Tile(id: 7, next: 8, prev: 6, accessible:{4}, landsOn: chute);

    function ladder = define landsOn(int playerID, int pieceID, int tileID) {
        print("Hooray! " | PlayerList[playerID].name | ", " | pieceID | " went up the ladder to tile " | TileList[tileID].accessible[0]);
        jump(playerID, pieceID, TileList[tileID].accessible[0]);
    }
}
make Tile(id: 1, next: 2, prev: 0, accessible:{5~9}, landsOn: ladder);

make Tile(id: 3, next: 4, prev: 2, accessible:{8}, landsOn: ladder);
make Tile(id: 6, next: 7, prev: 5, accessible:{0}, landsOn: chute);
make Tile(id: 7, next: 8, prev: 6, accessible:{4}, landsOn: chute);

function ladder = define landsOn(int playerId, int pieceID, int tileID) {
    print("Hooray! " | PlayerList[playerId].name | ", " | pieceID | " went up the ladder to tile " | TileList[tileID].accessible[0]);
    jump(playerID, pieceID, TileList[tileID].accessible[0]);
}
Board {
    NumTiles = 10;

    make Tile(id: 1, next: 2, prev: 0, accessible:{5~9}, landsOn: ladder);
    make Tile(id: 3, next: 4, prev: 2, accessible:{8}, landsOn: ladder);
    make Tile(id: 6, next: 7, prev: 5, accessible:{0}, landsOn: chute);
    make Tile(id: 7, next: 8, prev: 6, accessible:{4}, landsOn: chute);

    function ladder = define landsOn(int playerID, int pieceID, int tileID) {
        print("Hooray! " | PlayerList[playerID].name | ": " | pieceID | " went up the ladder to tile " | TileList[tileID].accessible[0]);
        jump(playerID, pieceID, TileList[tileID].accessible[0]);
    }

...
Board {
    NumTiles = 10;

    make Tile(id: 1, next: 2, prev: 0, accessible:{5~9}, landsOn: ladder);
    make Tile(id: 3, next: 4, prev: 2, accessible:{8}, landsOn: ladder);
    make Tile(id: 6, next: 7, prev: 5, accessible:{0}, landsOn: chute);
    make Tile(id: 7, next: 8, prev: 6, accessible:{4}, landsOn: chute);

    function ladder = define landsOn(int playerID, int pieceID, int tileID) {
        print("Hooray! " | PlayerList[playerID].name | ", " | "'s piece number " | pieceID | " went up the ladder to tile " | TileList[tileID].accessible[0]);
        jump(playerID, pieceID, TileList[tileID].accessible[0]);
    }
}

...
Overview
John Graham (System Architect)

Outline of a ROLL Program
Jesse Bentert (Tools & Language Guru)

ROLL Syntax
Daniel Wilkey (System Integrator)

Compiler Architecture & Testing
Yipeng Huang (Tester & Validator)

Conclusion & Demo
Lauren Pully (Project Manager)
Compiler Architecture

Front End

Back End

Global Variables

Board Implementation

Players Implementation

Dice Implementation

Next Turn Listener

Game

Player

Piece

Die

Deck

Card

Roller

Yacc

Flex

Roll.l

Roll.y

chutes.r

Deck

Player

Board Component

Window

Driver

Test
The Compiler Front End

```
%{
#define YYSTYPE
char*
#include <stdio.h>
#include "y.tab.h"
#include <stdlib.h>
#include <string.h>
char *makeString (char *temp);
char *LiteralToString (char *literal);
}%

roll.l

%{
#include <ctype.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#define YYSTYPE
#define null 0
void makeFile(char *fname, char *front, char *middle, char *back);
}%

roll.y

public class PlayersImplementation extends Players{
    public PlayersImplementation(Window gui){
        super(gui);
    }
    public void
    Global Variables.java

    public class PlayersImplementation extends Players{
        public PlayersImplementation(Window gui){
            super(gui);
        }
    }
    public void
    Board Implementation.java

    public class PlayersImplementation extends Players{
        public PlayersImplementation(Window gui){
            super(gui);
        }
    }
    public void
    Players Implementation.java

    public class PlayersImplementation extends Players{
        public PlayersImplementation(Window gui){
            super(gui);
        }
    }
    public void
    Dice Implementation.java

Game Demo {
    Players {
        GUI = 1;
        MaxPlayers = 6;
        MinPlayers = 2;
        NumPieces = 1;
        StartOn =
        {0, 0, 0, 0, 0, 0};
    }
    print("How many people are playing this game?");
}

chutes .roll

Flex

Yacc
```
1. **Overview**  
   John Graham (System Architect)

2. **Outline of a ROLL Program**  
   Jesse Bentert (Tools & Language Guru)

3. **ROLL Syntax**  
   Daniel Wilkey (System Integrator)

4. **Compiler Architecture & Testing**  
   Yipeng Huang (Tester & Validator)

5. **Conclusion & Demo**  
   Lauren Pully (Project Manager)
Environment and Tools

Flex + Bison Yacc
Conclusions

• Learned how to make a compiler using Flex and Bison Yacc

• Project management and planning

• Group collaborations