G!
A programming language for 2D games

Rachit Parikh rnp2102@columbia.edu
Divya Arora da2254@columbia.edu
Steve Lianoglou sl2585@columbia.edu
Amortya Ray ar2566@columbia.edu

Introduction

- Game Development – Tedious and complicated affair
- Lots of repetitive code and bookkeeping to ensure a proper functionality
- For example: check when 2 objects collide, check when a key is pressed, handle an event, …

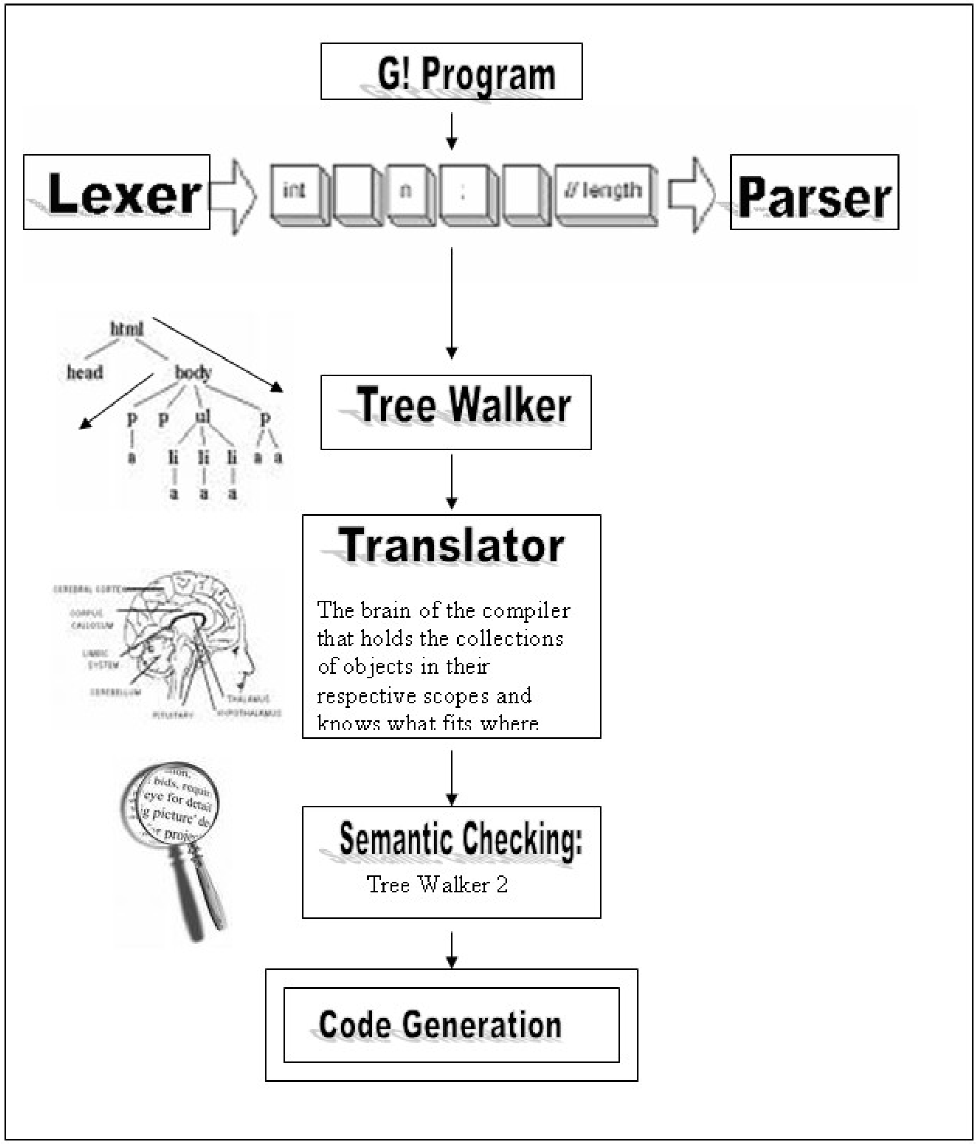
Motivation

- Hence G!
- Specifically for 2D games
- Allow the developer to focus on game play and target
- Bookkeeping handled at the backend
- Intuitive commands, minimal keywords, high flexibility

Implementation

- Based on the GTGE Library
- Library takes care of a lot of basic gaming functionality but it’s still Java – lengthy, redundant code

Compiler Structure:



The brain of the compiler that holds the collections of objects in their respective scopes and knows what fits where

Implementation

- G! Walker
  - Phase 1: Initializing symbol tables and other data structures
  - Phase 2: Type checking expressions, forward declarations of variables and functions
- G! Translator
  - Code Generation
  - Invoke javac compiler

G! v/s its Java Equivalent

- G! is free form, Java is not
- G! programs involve:
  - variable declarations and assignments
  - function definitions
  - if-else statements
  - while and for loops
  - an asynchronous statement type “when”

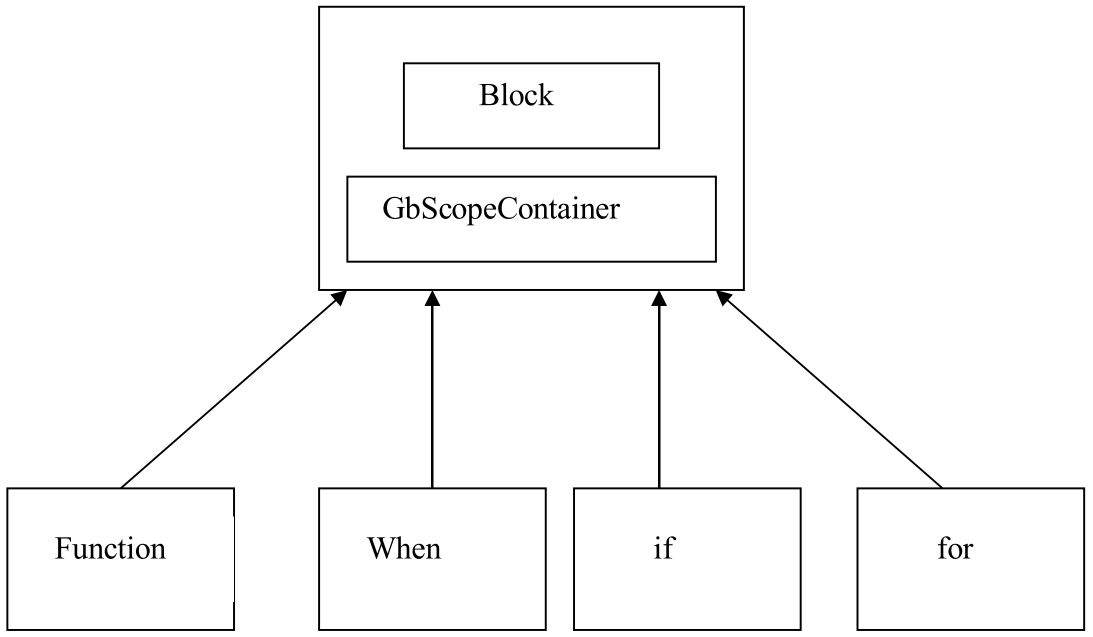
Its Java Equivalent

- Game class that includes:
  - Class level declarations
  - Initializations and setting the gamefield withing initResources()
  - An update method : the asynchronous event checks
  - A render method
  - Main method that launches game
  - Classes to handle collisions

Compiler Goals

- Find the collection of different statement types in the program
- Preserve the scope of each of these collections.
- Know what to do with each of these objects in the collection types
- Static/ semantic analysis of the program
- Generate a java equivalent

Our solution



Lessons Learnt

- Language development requires careful planning and analysis
- Before using any library, be sure to study it inside out
- Deal with the harder things first. Keep the simple stuff for later.
- Better time management to avoid the sleepless nights before submission!