스몰베이직 프로그램 디버거 설계 및 구현에 대한 연구

조문영 최광훈
전남대학교 전자컴퓨터공학부
jomy9649@gmail.com kwanghoon.choi@jnu.ac.kr

A Study on Design and Implementation of Debugger on Small Basic Programs

Munyoung Jo Kwanghoon Choi
Dept.Electronics and Computer Engineering, Chonnam National University, Gwangju

요 약

이 논문에서는 코딩 교육용 프로그래밍 언어 스몰베이직의 프로그램을 개발하는데 필요한 디버거를 설계 및 구현 하였다. 기존의 마이크로소프트 스몰베이직 프로그래밍 환경에서 디버거를 제공하지 않아 프로그램을 테스트하고 이해하는데 어려움이 있었다. 논문의 저자들이 참여한 공개 소프트웨어 기반 스몰베이직 프로그래밍 환경인 마이스몰베이직 상에서 디버거를 설계 및 구현하였다. 처음 코딩에 입문한 사람들이 스몰베이직 프로그램을 이해하고 디버그하는데 큰 도움이 됨을 확인하였다.

1. 서 론

다가오는 2018년부터 한국의 모든 중학교에서 코딩 교육이 의무화 된다. 하지만 정형화된 가이드라인이 없기 때문에 큰 우려의 목소리가 나오고 있다. 이에 따라 어떤 언어로 어떻게 교육을 할 것인지가 큰 문제가 되었다.

교육용 프로그래밍 언어는 하나의 대안으로 떠올랐다. 이것은 실제 어플리케이션을 개발하기 위한 목적이 아닌, 문제를 해결하는 과정을 통해 논리적이고 절차적인 사고력을 증진시키기 위한 언어이다. 스크래치나 파이썬 등 다양한 언어가 교육용 프로그래밍 언어로써 각광받고 있다. 그 중 마이크로소프트에서 만든 스몰베이직[1]은 해외에서 코딩 교육용 언어로 널리 활용되고 있다.

스몰베이직은 다양한 장점을 가진다. 텍스트 기반이라는 점에서 유아기의 아이들보다는 청소년기의 아이들에게 코딩교육을 하기에 적합한 환경을 제공하고, 다른 고급언어로의 진입장벽을 낮춰준다. 그리고 객체의 개념이 없고 최소한의 언어 요소만을 가지고 있기 때문에 초보자가 배우기에도 적합하다.

이러한 장점에도 불구하고 스몰베이직은 비공개 소프트웨어이기 때문에 여러 가지 어려움이 있다. 우선 교육을 위한 언어 확장과 라이브러리의 확장이 어렵다. 또한 닷넷 프레임워크에 종속되어 윈도우 운영체제에서만 실행이 가능하다. 무엇보다 프로그램을 디버깅하는데 쓰이는 디버거기능이 제공되지 않아서 많은 사용자들이 단순하게 출력하는 함수로 디버그를 흉내 내고 있을 따름이다.

이 논문에서는 특히 스몰베이직 프로그램 디버거가 없는 상황을 개선하기 위한 연구로 스몰베이직 프로그램을 디버깅할 수 있는 디버거를 설계 및 구현하였다.

2장 관련연구에서 마이크로소프트 스몰베이직 프로그래밍환경과 이 논문의 저자들이 개발 중인 오픈소스 소프트웨어 기반 스몰베이직 프로그래밍환경에 대해 설명한다. 3장에서 마이스몰베이직 프로그래밍 환경에서 디버거를 설계 및 구현한 내용을 상세하게 설명한다. 4장에서 디버거의 성능을 개선하는 사항을 논의하고, 5장에서 결론을 맺는다.

2. 관련연구

마이크로소프트의 스몰베이직[1]은 초보자가 쉽게 프로그램을 배울 수 있도록 만들어진 교육용 언어이다. 그래픽, 사운드, 타이머, 네트워크 등 흥미를 유발할 수 있는 라이브러리를 제공한다. 하지만 비공개 소프트웨어이기 때문에 코딩교육을 위한 다양한 라이브러리와 디버거와 같은 기능의 확장이 불가능하다.

이 논문의 저자는 스몰베이직 프로그래밍 환경을 위한 오픈소스 소프트웨어 프로젝트로 마이스몰베이직[2,3]을 개발하고 있다. 마이스몰베이직 프로그래밍 환경은 마이크로소프트 스몰베이직 환경과 호환 가능하도록 개발되었다. Java 기반으로 만들어져 윈도우즈 운영체제에 종속적이지 않다. 특히 오픈소스 기반이기 때문에 언어 확장, 라이브러리 확장, 프로그래밍 환경 확장이 가능하다. 마이스몰베이직 프로그래밍 환경 위에서 동작하는 디버거를 설계 및 개발한다.

3. 마이스몰베이직 디버거 설계 및 구현

먼저 마이스몰베이직 디버거를 사용자가 어떻게 활용하는지에 관하여 소개하고, 이 디버거의 상세 내부 구조에 대하여 설명한다.

3.1 사용자 관점의 마이스몰베이직 디버거

사용자는 마이스몰베이직의 디버거를 통해 디버깅 기능을 사용할 수 있다. 프로그램 실행 중 멈추고 싶은 지점에 중단점을 설정할 수 있고, 한 문장 실행(step), 다음 정지점까지의 실행(continue)을 통해 실행을 제어할 수 있다. 그리고 정지점에 도달하여 멈추게 되면 프로그램의 변수정보를 가져와 테이블로 보여준다.

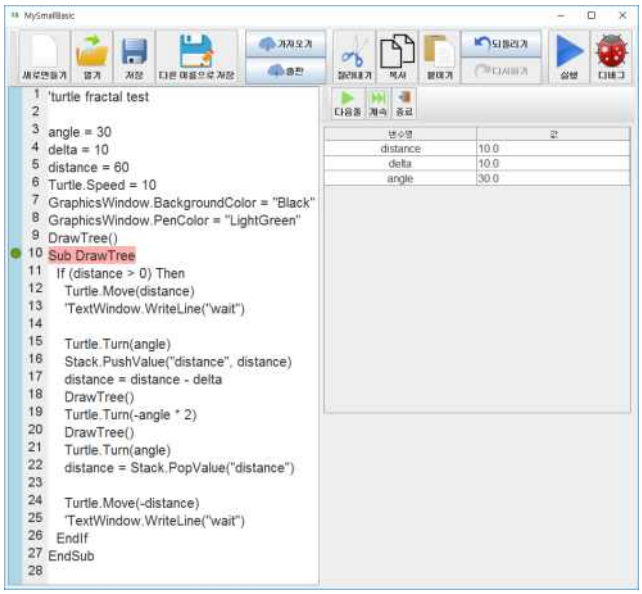

그림 1 마이스몰베이직 디버거 사용 예

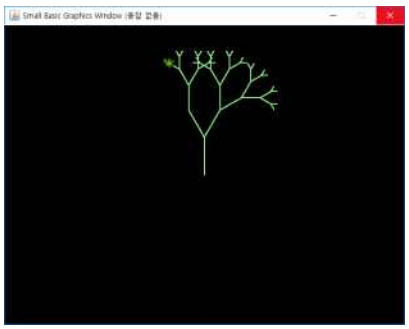

그림 2 프렉탈 트리 스몰베이직 프로그램 디버깅 예

그림1은 초보자가 이해하기 힘든 재귀함수 호출을 활용한 프로그램을 디버깅하는 사례를 보여준다. 그림1에서 보여주는 27라인의 프로그램은 재귀함수 호출 방법으로 그림2와 같은 트리를 그리는 프렉탈 프로그램이다. 재귀적 용법은 아주 중요하지만 프로그램을 처음 배우는 초보자들에게는 이해하기 힘든 개념이다. 마이스몰베이직의 디버거 기능을 사용한다면 재귀적인 개념을 쉽게 이해하는데 도움이 된다. 그림1과 같이 재귀적으로 불리는 함수의 시작점에 중단점을 설정한 후 다음 정지점까지 실행하는 버튼을 반복적으로 누르면 이 함수가 호출될 때마다 프로그램을 중단할 수 있다. 즉, 프렉탈 트리의 새로운 가지가 그려질 때 마다 프로그램을 멈추어 현재 프로그램 상태를 확인할 수 있다. 또한 오른쪽 패널에서 현재 변수들의 값을 확인함을 통해서 프로그램 진행 사항을 쉽게 알 수 있다. 프로그램 실행 과정을 더 상세하게 보고 싶다면 한 문장단위로 실행하는 다음 줄 버튼을 눌러서 한 문장 한 문장씩 디버깅 할 수도 있다.

3.2 마이스몰베이직 디버거 내부구조

마이스몰베이직[2,3]은 Java로 구현하였고 Java 플랫폼 디버거 아키텍처(JPDA)[4]를 활용하여 스몰베이직 프로그램 디버거를 구현하였다.

마이스몰베이직의 디버거의 구성은 그림3과 같다. 디버거 자체를 실행하는 자바 가상 기계와 디버깅 대상이 되는 스몰베이직 프로그램을 실행하는 자바가상기계가 별도의 프로세스에서 동작한다. 이 두 자바 가상기계는 JDWP(Java Debug Wire Protocol)를 통해서 디버깅 데이터를 주고받는다. 스몰베이직 프로그램 해석기의 실행 과정을 마이스몰베이직 IDE에서 제어하고 모니터링하기 위해서 자바가상기계 툴 인터페이스(JVM Tool Interface)가 필요하다.

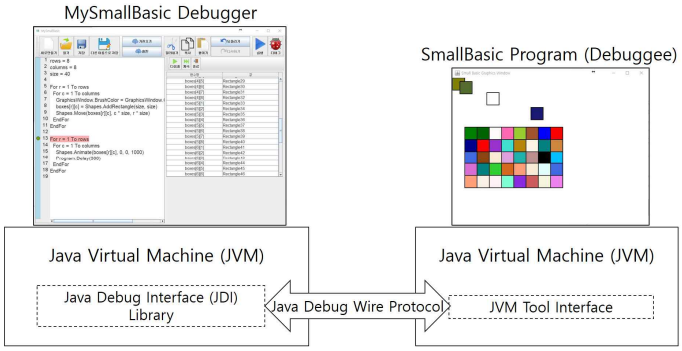

그림 3 마이스몰베이직 디버거 구성도

마이스몰베이직 디버거를 구현할 때 두 가지 사항을 중요하게 고려해야한다. 첫째 스몰베이직의 정지점을 구현하는 방법과 둘째 스몰베이직 프로그램의 변수 값을 가져와 보여주는 방법을 고려해야 한다.

마이스몰베이직의 파싱 단계에서 분석한 프로그램 각 문장의 줄 번호 정보를 Java 기반 해석기와 연결지어 마이스몰베이직 프로그램의 정지점을 설정할 수 있다. 마이스몰베이직의 각 문장이 실행될 때 불리는 Eval 메소드에 JDI를 통해 자바 수준의 정지점을 설정하고 각 문장을 실행할 때마다 이 문장에 정지점이 걸려있으면 실행을 정지시킨다.

그리고 JDI를 이용하여 실행중인 프로그램의 변수 환경 객체를 가져와 사용자에게 보여주게 된다. 다만 JDWP에서는 자바의 기본 타입과 String 객체만을 서로 다른 JVM사이에서 주고 받을 수 있기 때문에 디버깅 대상이 되는 해석기 프로그램의 변수 값을 가져올 때 임의의 클

래스 타입의 값을 읽어 올 수 없고 기본타입으로 표현된 값을 읽어오는 제약사항이 있다. 이를 가져와 IDE환경에서 변수이름과 값을 표로 나타내어 사용자에게 보여준다.

4. 논의사항

첫째 마이스몰베이직 디버거를 구현할 때 JDIScript[5]를 사용하여 복잡한 JDI 인터페이스를 쉽게 다루었다. JDIScript는 두 프로그램간의 연결 설정, 이벤트 등록 및 처리가 용이한 JDI 래퍼 인터페이스이다. 마이스몰베이직 환경의 디버거는 궁극적으로 JDI(Java Debug Interface)를 통해서 Java 프로그램 수준에서 디버깅 대상 프로그램의 실행을 제어하고 모니터링 할 수 있다.

둘째 스몰베이직의 각 문장이 실행될 때 마다 중단점이 걸려있는지 검사하는 오버헤드가 존재하여 디버깅 속도가 느려질 수 있다. 예를 들어 프렉탈 프로그램은 만 번 이상 루프를 돌면서 점을 한 번씩 찍어 그림4와 같이 기하학적인 모양을 그리는 예제이다. 디버거를 실행시키지 않고 프로그램을 실행시키면 빠르게 점이 찍혀 결과를 확인할 수 있다. 반면, 중단점을 설정하지 않고 디버거를 실행시키면 점이 찍히는 과정을 확인할 수 있을 정도로 느려지게 된다.

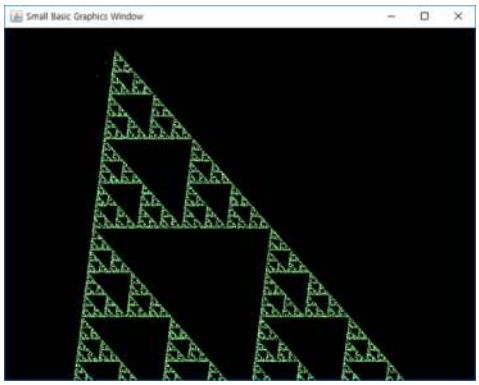

그림 4 디버깅의 오버헤드를 확인할 수 있는 프렉탈 프로그램

교육용 언어의 주된 사용자인 초보 프로그래머들의 간단한 프로그램을 디버깅하기에는 문제가 없지만 게임과 같은 복잡한 프로그램을 디버깅할 때 속도가 느린 어려움이 있을 수 있다. 이 문제는 중단점이 아닌 모든 구문에 대해서도 조건검사를 하기 때문에 발생한다. 향후 중단점을 건 문장을 실행할 때만 자바 디버거가 멈출 수 있도록 연구할 예정이다. 이렇게 효율적으로 조건검사를 하도록 디버거 구조를 변경하면 디버깅으로 인한 오버헤드가 많이 줄어들 것으로 보인다.

5. 결 론

본 연구에서는 교육용 언어 스몰베이직 프로그래밍 환경인 마이스몰베이직을 확장하여 디버거를 설계 및 구현하였다. 이는 기존의 마이크로소프트의 스몰베이직이 디버깅 기능을 제공하지 않아 어려움을 겪었던 많은 초보 프로그래머들에게 스몰베이직 프로그램을 더 잘 이해할 수 있는 환경을 제공할 수 있을 것이다.

향후연구로 4장에서 논의한 디버거의 성능 개선에 관한 연구를 진행할 계획이다.

6. 참고문헌

[1] Microsoft Small Basic, http://smallbasic.com.

[2] MySmallBasic, https://github.com/kwanghoon/MySmallBasic.

[3] 김가영,정승완,김태진,조영민,김범준,최광훈, 스몰베이직 프로그램 해석기 설계 및 구현에 관한 연구, 정보과학회 동계학술대회, 학부생 포스터, 2016년12월.

[4] Java Platform Debugger Architecture (JPDA), https://docs.oracle.com/javase/7/docs/technotes/guides/jpda/

[5] JDIScript: A Simple Scripting Frontend for the Java Debugger Interface, https://github.com/jfager/jdiscript/