공학 석사학위논문

코딩 교육을 위한 프로그래밍 언어 환경 설계 및 구현

전남대학교 대학원
전자컴퓨터공학과

김 가 영

2019년 8월

코딩 교육을 위한 프로그래밍 언어 환경 설계 및 구현

이 논문을 공학 석사학위 논문으로 제출함

전남대학교 대학원
전자컴퓨터공학과

김 가 영

지도교수 최 광 훈

김가영의 공학 석사의 학위논문을 인준함

심사위원장 전남대학교 교 수 공학박사 박 혁 로 (인)

심사위원 전남대학교 부교수 공학박사 김 경 백 (인)

심사위원 전남대학교 부교수 공학박사 최 광 훈 (인)

2019년 8월

목 차

그 림 목 차

표 목 차

코딩 교육을 위한 프로그래밍 언어 환경 설계 및 구현

김 가 영

전남대학교 대학원 전자컴퓨터공학과

(지도교수 : 최 광 훈)

(국문초록)

본 논문에서 오픈소스 소프트웨어 기반 교육용 스몰베이직 코딩 환경을 개발한 내용을 소개한다. 소프트웨어의 역할이 중요해진 시대에 초중고교에서는 일정시간 이상의 코딩 교육을 의무화하고 있으며, 대학에서 타 전공학생에게 코딩교육을 하고 있는 것이 세계적인 추세이다. 코딩 교육에 사용될 언어가 주목 받고 있는 가운데, 스몰베이직은 코딩 입문자를 위한 간단하고 배우기 쉬운 텍스트 기반 프로그래밍 언어이다. 하지만 기존의 마이크로소프트 스몰베이직 환경은 비공개 소프트웨어로 환경을 확장하거나 추가하기 어렵다는 단점을 가지고 있다. 따라서 스몰베이직과 같은 입문자에게 간단하고 쉬운 환경을 제공하면서도 언어 및 라이브러리의 확장이 가능한 교육용 프로그래밍 언어에 대한 연구가 필요하다.

본 논문에서는 마이크로소프트 스몰베이직을 윈도우, 리눅스, 맥에서 모두 사용할 수 있도록 Java로 확장한 새로운 스몰베이직 코딩 환경 마이스몰베이직을 제안한다. 기존 스몰베이직 언어에서 사용된 프로그램과의 호환성을 제공하며 오픈소스 프로젝트이기 때문에 누구나 개발에 참여할 수 있다. 또한 이 연구를 통해 이전에 문서화되지 않았던 스몰베이직 언어의 파서와 동적 타입 변환에 대한 명세를 처음으로 작성하였다.

본 논문에서 제안한 마이스몰베이직은 PC 기반의 다양한 운영체제에서 사용이 가능해졌다. 입문자에게 흥미를 일으킬만한 라이브러리의 추가가 자유로워졌으며, 디버깅 기능을 추가함으로써 프로그램 구조에 대한 이해를 더욱 높여준다.

제 1 장 서론

현 사회는 정보화 사회를 지나 소프트웨어 중심 사회로 불리고 있다. 소프트웨어 중심 사회란 사회의 모든 영역에서 생산성을 향상시키기 위해 소프트웨어를 도입하는 사회를 말한다. 이에 따라 우리는 모든 영역에서 사용되는 소프트웨어에 대해 잘 이해해야 하며 더 나아가서 새로운 소프트웨어를 구성하는 능력이 필요하다. 이러한 추세에 맞춰 초중고교에서 코딩 교육이 의무화되고 있으며, 대학에서 타 전공의 학생들도 코딩 교육에 적극 참여하고 있다.

하지만 이 코딩 교육을 어떤 언어로 진행할 것인지가 하나의 문제가 되었다. 교육용 프로그래밍 언어는 실제 프로그램을 개발하기 위한 목적의 언어가 아닌, 문제를 논리적이고 절차적으로 해결할 수 있는 능력을 키우기 위한 언어이다. 현재 스크래치(Scratch)[1]와 앱 인벤터(App Inventor)[2], 파이썬(Python)[3]은 교육용 프로그래밍 언어로 각광받고 있으며, 해외에서는 마이크로소프트의 스몰베이직(Small Basic)[1,2]이 교육용 프로그래밍 언어로 활용되고 있다.

마이크로소프트의 비제에 라지(Vijaye Raji)가 개발한 스몰베이직은 코딩을 처음 배우는 입문자를 위한 프로그래밍 언어이며 윈도우 기반 코딩 환경을 갖추고 있다. 스몰베이직은 최소한의 언어 요소만을 가지고 있으며, 코딩 환경의 메뉴가 단순하게 구성되어 있어 입문자가 코딩을 배우기 적합한 환경을 제공한다. 또한 라이브러리를 활용하여 텍스트 및 그래픽스 프로그램을 간단하게 작성할 수 있고, 소스 프로그램을 공유할 수 있는 일종의 앱 스토어를 제공한다.

스몰베이직 이외의 다른 교육용 코딩환경이 있다. 스크래치와 앱 인벤터는 그래픽 기반의 코딩 환경을 제공하여 사용자가 블록으로 된 프로그램 조각을 쌓아 프로그램을 구성하고 실행할 수 있다. 전 세계에서 코딩 교육에 많이 활용되고 있으며, 초등학생뿐만 아니라 처음 코딩에 입문하는 어른에게도 유용하게 사용되고 있다. 파이썬은 텍스트 기반의 환경과 풍부한 라이브러리를 제공하여 사용자가 다양한 프로그램을 작성할 수 있다. 특히 데이터 사이언스와 머신러닝 분야의 라이브러리가 많아 이와 관련된 응용 분야를 염두에 두고 있다면 파이썬을 선택하는 것이 좋다.

---

1) Scratch, https://scratch.mit.edu
2) App Inventor, http://appinventor.mit.edu
3) Python, http://www.python.org

스몰베이직과 다른 교육용 코딩 언어를 다음과 같이 비교할 수 있다. 스크래치와 앱 인벤터 같은 교육용 코딩환경을 통해서 코딩의 기초 개념을 배울 수 있지만, 그림 기반 코딩환경이기 때문에 다양한 라이브러리를 이용할 수 있는 텍스트 기반 코딩환경에 비해 프로그래밍 경험이 제한적이다. 반면에 파이썬은 텍스트 기반 코딩환경을 제공함으로써 다양한 프로그래밍 경험을 쌓을 수 있지만, 입문자가 사용하기에는 너무 많은 특징을 가지고 있어 오히려 큰 장벽이 된다. 이와 달리, 스몰베이직은 텍스트 기반 코딩환경을 제공하여 다른 고급 언어로의 진입장벽을 낮춰준다. 또한, 최소한의 개념만 도입되어 설계되었기 때문에 처음 코딩을 배우는 입문자에게 적합하다.

입문자가 코딩을 배우기 쉬운 환경을 제공함에도 불구하고 스몰베이직을 이용해 코딩 교육을 진행하는데 있어 세 가지 문제점이 있다. 첫째, 마이크로소프트에서 스몰베이직 환경을 닷넷 프레임워크에 의존적으로 개발하였기 때문에 리눅스, 맥 운영체제, 웹, 안드로이드 환경에서 실행할 수 없다. 둘째, 스몰베이직은 비공개 소프트웨어로 코딩 교육에 흥미를 높일 수 있는 라이브러리를 추가할 수 없으며, 구조 또한 공개되어 있지 않다. 셋째, 마이크로소프트 스몰베이직 환경은 실행 바이너리 파일만 공개되어 있고 소스 파일에 접근할 수 없기 때문에 새로운 기능을 추가할 수 없다. 마이크로소프트 스몰베이직은 디버깅 기능이 없어 교육자가 이러한 기능을 추가하려 해도 외부에서 스몰베이직에 기능을 개발하거나 확장할 수 없다.

이 논문의 연구 목표는 스몰베이직 프로그래밍 언어의 코딩 교육 관점에서의 장점을 유지하며 앞에서 언급한 마이크로소프트 스몰베이직의 문제점들을 개선한 새로운 스몰베이직 기반 코딩 교육용 플랫폼을 개발하는 것이다. 이 연구 목표를 달성하기 위해 오픈소스 소프트웨어 프로젝트 기반으로 특정 운영체제에 종속되지 않는 스몰베이직 코딩 환경과 누구나 자유롭게 참여하여 새로운 라이브러리나 기능을 추가할 수 있도록 개발할 필요가 있다. 기본적으로 마이크로소프트 스몰베이직에서 실행 가능한 프로그램을 동일하게 동작하도록 호환성을 지원하는 해석기(Interpreter)와 표준 라이브러리를 개발하였다. Java 기반으로 개발하여 윈도우 운영체제뿐만 아니라 리눅스와 맥 운영체제에서 동일하게 동작하여 기존의 운영체제 종속성을 없앴다. 또한 기존 스몰베이직 환경에서 제공하지 않았던 햄스터 로봇 라이브러리, Facebook/Twitter 라이브러리, SQLite 데이터베이스 라이브러리 Weka[3] 기반 학습 라이브러리를 추가함으로써 코딩 교육의 흥미를 유도할 수 있었다. 그리고 스몰베이직 프로그램 디버거 기능을 추가하여 초보자가 실행과정을 따라가며 제어 흐름을 살피고 변수 값이 변화

되는 과정을 확인할 수 있었다.

오픈소스 소프트웨어 기반 스몰베이직 코딩 환경을 개발하며 겪었던 기술적 문제점은 다음과 같다. 스몰베이직 파서와 해석기를 개발할 때 스몰베이직 언어에 대한 구문과 의미를 정의가 필요하다. 그러나 이를 정의하는 마이크로소프트 스몰베이직 언어의 공식 문서가 없다. 이를 해결하기 위해, 기존 마이크로소프트 스몰베이직 코딩 환경과 기존에 작성된 스몰베이직 프로그램들을 모아 역공학 분석하여 파서와 해석기의 명세를 작성하였다. 이렇게 분석한 내용을 이 논문에서 요약 정리한다.

이 연구에서 기여한 바는 다음과 같다.

- 오픈소스 소프트웨어로 교육용 코딩환경 마이스몰베이직[4,5]과 표준 라이브러리를 개발하였다. 자바를 이용하여 개발함으로써 특정 운영체제에 종속되지 않는 스몰베이직 코딩환경을 개발하였다.
- 기존 마이크로소프트 코딩 환경에서 개발한 스몰베이직 프로그램의 호환성을 보장한다. 스몰베이직 튜토리얼 문서에서 추출한 59개의 프로그램과 4개의 복잡한 프로그램을 실행 및 비교 테스트하여 동일하게 동작함을 확인하였다.
- 스몰베이직 프로그래밍 언어의 상세 명세를 처음으로 문서화하였다. 스몰베이직의 파서와 렉서를 구현함으로써, 스몰베이직의 문법을 문서화하였으며, 스몰베이직이 채택하고 있는 동적 형 변환 규칙을 분석하고 이에 대해서도 문서화하였다.
- 오픈소스 소프트웨어로써의 마이스몰베이직 라이브러리 아키텍처를 문서화하였다. 이 라이브러리 아키텍처를 이용하여 확장 라이브러리의 개발을 더 용이하게 한다.
- 스몰베이직 개발문서에 소개되지 않았던 그래픽스 아키텍처를 문서화하고 개발하였다. 다양한 그래픽스 라이브러리의 구조를 파악함으로써 스몰베이직 그래픽스와의 동일성을 더 보장할 수 있다.
- 기존 마이크로소프트 스몰베이직에 없던 디버거를 개발하였다. 디버거를 이용하여 처음 코딩에 입문한 사람들이 프로그램을 이해하고 디버깅할 수 있다.
- 기존 스몰베이직에서 제공하지 않던 라이브러리를 개발하였다. 코딩 교육으로써 입문자의 흥미를 높일 수 있는 다양한 라이브러리를 만들어 제공하며, 오픈소스 소프트웨어로 추가하고자 하는 라이브러리를 개발하여 사용할 수 있다.
- 마이스몰베이직을 강의에 적용하여 교육용 프로그래밍 언어로써 적합한지에 대해서 확인하였다. 학생들이 사용해봄으로써 교육용 언어로써의 스몰베이직을 분석하

고 이에 따른 효과를 확인하였다.

본 논문의 구성은 다음과 같다. 2장 관련연구에서 마이크로소프트 스몰베이직을 소개하고 기존 교육용 프로그래밍 언어와 비교한다. 3장에서 오픈소스 소프트웨어 프로젝트 마이스몰베이직의 해석기에 대해서 설명한다. 4장에서 해석기의 확장과 코딩 교육 강의에 적용한 사례를 설명하고, 5장에서 결론을 맺고 향후 연구에 대해서 설명한다.

제 2 장 관련 연구

제 1 절 마이크로소프트 스몰베이직

마이크로소프트의 비제에 라지(Vijaye Raji)가 개발한 스몰베이직(Small Basic)은 처음 코딩을 접하는 사람을 위한 교육용 프로그래밍 언어이고, 현재 윈도우 운영체제에 코딩 환경을 제공한다. 프로그래밍 언어가 간단하고, 코딩환경도 매우 단순하여 적응하기 쉽고, 다양한 라이브러리를 활용하여 프로그램을 쉽게 작성할 수 있다. 예를 들어, 스몰베이직을 이용해 플리커 웹사이트에서 지정한 주제의 사진들을 찾아 슬라이드 쇼하는 프로그램을 (그림 1)과 같이 간단하게 작성할 수 있다.

```
01: For i = 1 To 20
02:   pic = Flickr.GetRandomPicture(“Korea”)
03:   GraphicsWindow.DrawResizedImage(pic, 0, 0, 640, 480)
04: EndFor
```

그림 1 스몰베이직 프로그램 예제: 슬라이드 쇼

다음 (그림 2)는 마우스로 클릭한 위치에 랜덤한 색상의 원을 그리는 스몰베이직 프로그램이다. 마우스 클릭에 대한 이벤트가 발생했을 때 실행해야 할 코드는 7번째 줄에 작성된 함수로 정의한다. 마우스 클릭에 대한 이벤트와 이벤트가 발생했을 때 실행할 함수와의 연결은 5번째 줄에 작성된 코드와 같다.

```
01: GraphicsWindow.BrushColor = "Blue"
02: preColor = GraphicsWindow.BrushColor
03: curColor = GraphicsWindow.BrushColor
04:
05: GraphicsWindow.MouseDown = OnMouseDown
06:
07: Sub OnMouseDown
```

```
08:   x = GraphicsWindow.MouseX - 10
09:   y = GraphicsWindow.MouseY - 10
10:
11:   ChangeColor:
12:     curColor = GraphicsWindow.GetRandomColor()
13:
14:     If curColor = preColor Then
15:       Goto ChangeColor
16:     Else
17:       preColor = curColor
18:       GraphicsWindow.BrushColor = curColor
19:     EndIf
20:
21:   GraphicsWindow.FillEllipse(x, y, 20, 20)
22: EndSub
```

그림 2 스몰베이직 프로그램 예제: 임의의 색상으로 원 그리기

스몰베이직 프로그래밍 언어의 특징은 다음과 같다.

- 스몰베이직의 키워드는 전체 16개, If, Then, ElseIf, Else, EndIf, While, EndWhile, For, To, Step, EndFor, Goto, Sub, EndSub, And, Or이다.
- 특별한 선언 없이 변수를 사용하고, 스코프(Scope) 개념 없이 모든 변수를 전역 변수로 취급한다. 변수의 초기 값은 빈 문자열 “”이다.
- 숫자, 문자열, 부울, 배열을 지원한다. 모든 값은 문자열로 변환할 수 있고, 이러한 문자열을 다시 숫자, 부울, 배열로 되돌릴 수 있다. 3장에서 자세히 설명한다.
- 분기문 Goto와 서브루틴, 조건문 If, 반복문 For와 While을 지원한다. 서브루틴은 인자를 전달하고 값을 반환하는 방법을 제공하지 않는다. 문장으로서 서브루틴 호출만을 허용하고, 식으로써 사용할 수 없다.
- 라이브러리에서 함수와 변수를 제공한다. 서브루틴과 달리 함수는 인자를 전달하고 값을 반환하는 방법을 제공하여 식으로써 함수 호출을 허용한다. 그리고 라이

브러리에서 정의한 변수를 읽고 쓸 수 있다.

스몰베이직은 (표 1)에 나열한 20개의 표준 라이브러리를 제공한다. 텍스트와 그래픽 입출력을 비롯하여 코딩 교육의 흥미를 높일 수 있는 다양한 라이브러리를 갖추고 있다.

표 1 스몰베이직 표준 라이브러리

라이브러리	설명
Array	Array functions
Clock	System clock functions
Controls	Button, TextField, MultiLine TextField
Desktop	Desktop wallpaper
Dictionary	Online dictionary
Flickr	Access to Flickr photo service
File	File and directory management
GraphicsWindow	Graphics functions and variables
ImageList	Image management
Math	Math functions
Mouse	Mouse cursor and button properties
Network	File download
Program	Program execution controls
Shapes	Movable graphics figures
Sound	Sound play, stop, pause functions
Stack	Stack functions
TextWindow	Text-console based input/output
Text	Text manipulation
Timer	Timer functions
Turtle	Turtle graphics

스몰베이직 프로그래밍 언어로 코딩할 수 있는 환경은 (그림 3)의 마이크로소프트에서 제공하는 코딩 환경이 유일했었다. 이 코딩 환경은 파일 열기와 저장, 편집 복사와 붙여넣기, 프로그램 실행과 같이 반드시 필요하고 이해하기 쉬운 메뉴들만 포함되어 있다. 따라서 코딩 입문자가 쉽게 사용할 수 있다. 그리고 편집 창에서 자동 완성

기능을 제공하여 라이브러리 함수와 변수의 이름을 기억하지 못하더라도 앞부분만 일부 작성하면 일치하는 함수와 변수 목록을 보여준다. 마지막으로, 스몰베이직 앱스토어에서 소스 코드를 가져오거나 자신이 작성한 소스 코드를 출판하는 메뉴를 제공한다. 스몰베이직 코딩 커뮤니티를 통해 서로 배울 수 있다.

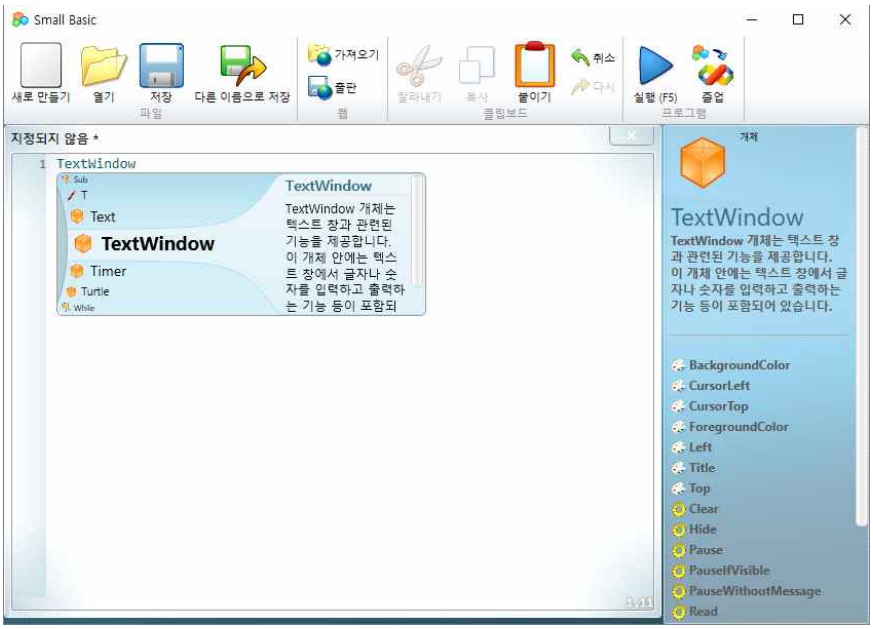


그림 3 마이크로소프트 스몰베이직 코딩 환경

제 2 절 코딩 교육을 위한 다른 프로그래밍 언어와 비교

교육용 프로그래밍 언어와 환경은 크게 두 가지로 구분할 수 있다. 첫째, 그림 기반 코딩 환경이 있다. 그래픽 환경에서 블록 그림으로 표현된 프로그램 조각을 조합하여 프로그램을 구성하는 방식으로, 스크래치와 앱 인벤터가 대표적인 사례이다. 특별한 사전 지식을 요구하지 않고 흥미를 높일 수 있는 요소로 구성되어 초등학생 수준을 대상으로 하는 코딩 교육에 활발히 사용되고 있다. 둘째, 텍스트 기반 코딩 환경이 있다. 특히 풍부한 라이브러리를 기반으로 다양한 프로그램을 작성할 수 있는 파이썬은 교육용 프로그래밍 언어로 많이 활용되고 있다. 데이터 사이언스와 머신 러닝 분야의 커뮤니티에서 개발한 방대한 라이브러리를 갖추고 있기 때문에 이 분야를 목표로 한다면 교육용 코딩 언어로 파이썬을 선택할 수 있을 것이다.

(표 2)에서 앞서 설명한 두 가지 코딩 교육용 프로그래밍 언어와 스몰베이직을 비교

한 내용을 정리하였다.

표 2 코딩 교육용 프로그래밍 언어 비교

	단순성	이용성	라이브러리
스크래치, 앱 인벤터	Simple (Codeless)	Web, PC, Android	Limited
마이크로소프트 스몰베이직	Simple	Windows	Limited
파이썬, C/C++/Java	Complex	OS-neutral	Rich

스몰베이직 언어를 다른 코딩 교육용 프로그래밍 언어 및 환경과 비교하면, 그림 기반 코딩 환경에 비해 텍스트 기반 프로그래밍 언어이며 다른 텍스트 기반 프로그래밍 언어에 비해 최소의 언어 특징으로 간단하게 설계되어 초보자가 처음 코딩 개념을 배우기에 적합한 장점이 있다. 그러나 현재 마이크로소프트 스몰베이직의 유일한 코딩 환경은 윈도우에서만 실행 가능하고 소스 코드를 비롯한 내부 구조를 명시적으로 공개하지 않아 새로운 기능을 개발할 때 특정 회사 개발 그룹에 의존적인 단점이 있다.

대부분의 프로그래밍 언어들의 공통 속성인 텍스트기반 언어를 선택한다면 스몰베이직의 장점이 분명하다. 파이썬을 비롯한 C/C++/Java는 클래스, 객체, 포인터 등과 같은 초보자의 코딩 교육에 필요한 것 이상의 언어적 특징을 보유하고 있어 배움에 오히려 장벽이 될 수 있다. 반면에 스몰베이직은 최소한의 언어 특징만을 갖추도록 설계되어 쉽게 코딩을 배울 수 있다. 스몰베이직을 교육용 프로그래밍 환경으로 선택하는데 장애가 되는 요소는 (표 2)에서 비교한 것과 같이 윈도우 운영체제에서만 코딩 교육을 진행할 수 있다는 점과 다양한 라이브러리와 코딩 환경에 필요한 새로운 기능을 추가하는 것이 불가능하다.

이 연구의 목표는 마이크로소프트 스몰베이직에서 정의한 프로그래밍 언어의 구문과 의미를 호환되도록 유지하여 배우기 쉬운 언어의 장점을 유지하면서 오픈소스 소프트웨어 프로젝트 커뮤니티의 기여를 통해 앞서 언급한 두 가지 단점을 극복하는 것이다.

제 3 장 오픈소스 소프트웨어 기반 스몰베이직 코딩 환경

이 장은 본 연구에서 개발한 오픈소스 소프트웨어 기반 스몰베이직 코딩 환경 프로젝트, 마이스몰베이직(MySmallBasic)에 대하여 설명한다.

제 1 절 오픈소스 프로젝트 마이스몰베이직의 개요

오픈소스 프로젝트 마이스몰베이직은 자바로 작성한 스몰베이직 해석기, 표준 및 확장 라이브러리, 디버거를 포함한 편집 및 실행 환경으로 구성되어 있다. (그림 4)은 마이스몰베이직을 활용하여 테트리스 프로그램을 실행하는 화면이다.

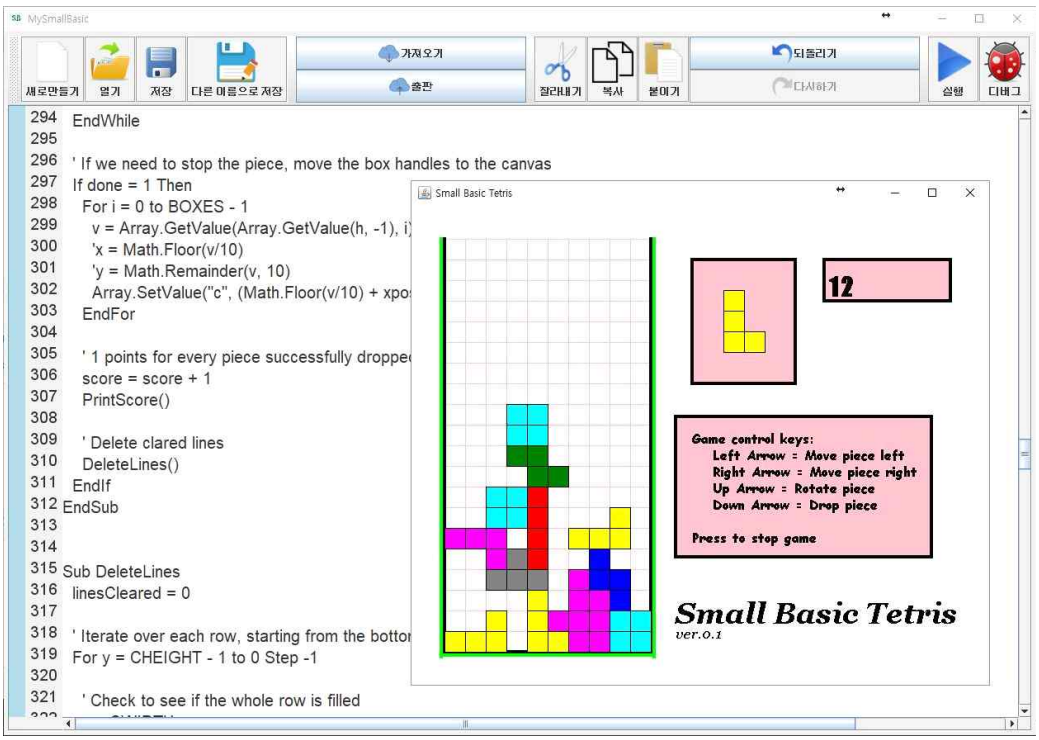


그림 4 마이스몰베이직 코딩 환경

아래 깃허브(GitHub) 웹 사이트에 이 프로젝트의 소스코드, 문서, 스몰베이직 예제 프로그램들을 모두 내려 받을 수 있다.

https://github.com/kwanghoon/MySmallBasic

마이스몰베이직 프로젝트는 총 2만여 라인의 자바 소스 프로그램으로, 스몰베이직 프로그램을 실행하는 해석기가 7,448라인, 스몰베이직 라이브러리가 13,445라인 규모에 해당한다.

이 오픈소스 프로젝트에 현재까지 15명의 개발자가 참여하여 소스 프로그램을 기여했다. 최광훈 교수의 지도하에 연세대의 3명(김태진, 조영민, 김범준)과 전남대 2명(김가영, 정승완)이 스몰베이직 해석기를 구현하였다. 그리고 전남대 6명(김가영, 김지용, 박세영, 정승완, 조문영, 조성모)이 스몰베이직의 표준 및 확장 라이브러리를 구현하였으며, 디버거를 포함한 GUI 코딩환경을 개발하였다. 베트남 HUST 대학의 6명이 확장 라이브러리를 개발하여 이 프로젝트에 기여하였다. 개발 파트에 대한 상세한 내용은 부록에서 확인할 수 있다.

마이스몰베이직 프로젝트는 GPL(General Public License)를 채택하여 관련 소스 프로그램을 수정하였을 경우 공개하도록 의무화하였다. 그 이유는 마이스몰베이직 프로젝트를 누구나 다운받아 수정 가능하지만 기존의 스몰베이직 프로그램 동작과 다르게 변형된 프로젝트가 확산되는 것을 최대한 방지하기 위함이다. 2008년 처음 마이크로소프트 스몰베이직 코딩 환경이 공개된 이후로 상당히 큰 커뮤니티가 형성되었고 이 커뮤니티에서 상당한 양의 스몰베이직 프로그램들을 작성해왔다. 이러한 라이선스 정책을 통해 호환성을 유지하여 기존의 스몰베이직 코딩 커뮤니티가 활용할 수 있는 프로젝트로 진행하고자 한다.

특히 스몰베이직 코딩 환경 간 호환성을 유지하기 위해 마이크로소프트에서 발행한 스몰베이직 튜토리얼에 나열된 59개 예제 프로그램들을 활용하였다. 이 프로그램들은 최대 36라인, 전체 합계 470라인으로 비록 개별적으로 크기가 작지만 스몰베이직 프로그래밍 언어의 각 기능 (If, For, While 등)을 검증하고 표준 라이브러리 함수들을 정확히 구현하였는지 확인하는데 충분하다. 이 예제 프로그램들을 마이스몰베이직에서 실행한 결과와 마이크로소프트 프로그래밍 환경에서 실행한 결과가 모두 동일한 것을 확인하였다.

표 3 스몰베이직 호환성 확인을 위한 벤치마크 프로그램

프로그램 명	라인 수	제공하는 기능
01_HelloWorld	1	콘솔창에 문자열 출력
02_FontYellowColor	2	지정된 색상으로 콘솔창에 문자열 출력
03_Variables	3	콘솔창을 통한 사용자의 입력
04_Temperature	4	콘솔창을 통한 숫자입력, 수식 계산
05_If	6	If 조건에 따른 문장 실행
06_Goto	10	Goto문
07_For	3	For문
08_ForStep	3	지정된 Step만큼 계산되어 For문
09_While	5	While문
10_GraphicWindow	1	그래픽 창 띄우기
11_GraphicWindowConfig	5	지정한 설정으로 그래픽 창 띄우기
12_DrawLine	4	그래픽 창에 선 그리기
13_LineColor	6	지정한 색으로 그래픽 창에 선 그리기
14_LineThickness	7	설정한 두께로 그래픽 창에 선 그리기
15_Rectangle	6	그래픽 창에 사각형 그리기
16_Ellipse	6	그래픽 창에 타원형 그리기
17_Circle	6	그래픽 창에 원 그리기
18_Random	7	랜덤 색상, 위치로 그래픽 창에 원 그리기
19_Fractal	23	난수 생성, 삼각형 fractal 그리기
20_Subroutine	8	함수 정의 및 함수 호출
21_Array	9	배열에 값 저장 및 출력
22_ArrayIndex	11	배열의 문자열 인덱스 처리
23_MultiDimArray	14	2차원 배열에 대한
24_Event	7	마우스 이벤트 처리
25_Events	13	마우스 클릭, 키보드 누름 이벤트 처리
26_Flickr	6	마우스 클릭할 때마다 사진 변경

스몰베이직에서 제공하는 튜토리얼 문서에 작성된 프로그램들[6]과 함께 스몰베이직 커뮤니티에서 개발한 최대 천 라인 규모의 큰 예제 프로그램도 프로젝트에 포함시켜 테스트하였다. (표 3)은 스몰베이직에서 제공하는 프로그램 26개를 정리한 것이며, (표 4)는 천 라인 규모의 큰 예제 프로그램 4가지를 사례를 보여준다. 각 프로그램에

서 제공하는 기능이 마이스몰베이직에서 정상적으로 동작하는지 확인하고, 스몰베이직과 마이스몰베이직의 동작 결과가 동일한지 확인하였다.

표 4 스몰베이직 호환성 확인을 위한 복잡한 예제 프로그램

프로그램 명	라인 수	프로그램 설명
Bricks	243	Bricks game
CollisionPhysics	447	Physics simulation
Tetris	536	Tetris game
Sokoban	1165	Sokoban game

제 2 절 스몰베이직 해석기

오픈소스 프로젝트 마이스몰베이직은 자바로 작성한 해석기를 통해 스몰베이직 프로그램을 실행한다. 스몰베이직 프로그래밍 언어의 구문과 의미를 엄밀하게 설명하는 문서가 없기 때문에, 마이크로소프트 스몰베이직 코딩 환경에서 실행해보면서 그 구문과 의미를 역공학(Reverse Engineering) 분석하였다. 역공학 분석하여 만들어진 해석기는 (그림 5)와 같이 파싱 단계, 기본 블록으로 변환하는 단계, 실행하는 단계 순서로 실행된다.

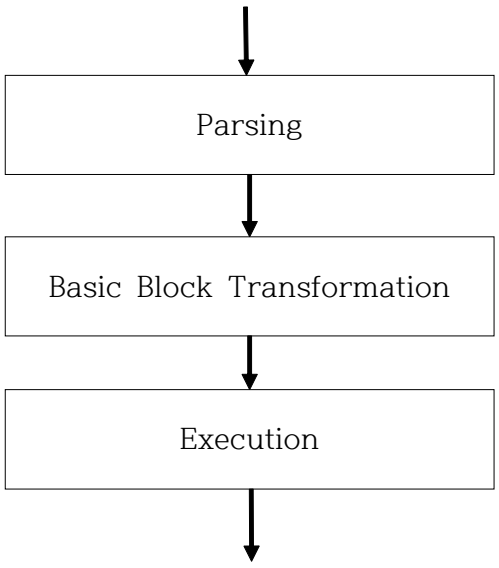


그림 5 해석기 실행 단계

이 절에서는 스몰베이직 해석기를 역공학 분석하여 만든 추상구문 트리의 구현과 해석기의 실행 단계인 파싱 단계, 기본 블록 변환 단계, 실행 단계를 차례로 설명한다.

2.1 스몰베이직 해석기 추상 구문 트리

(그림 6)는 역공학 분석하여 만든 해석기의 추상구문트리를 보여준다. 스몰베이직 프로그램은 문장(Statement)을 표현하는 클래스 11개, 식(Expression)을 표현하는 클래스 11개, 값(Value)을 표현하는 클래스 4개의 클래스로 이루어져 있다.

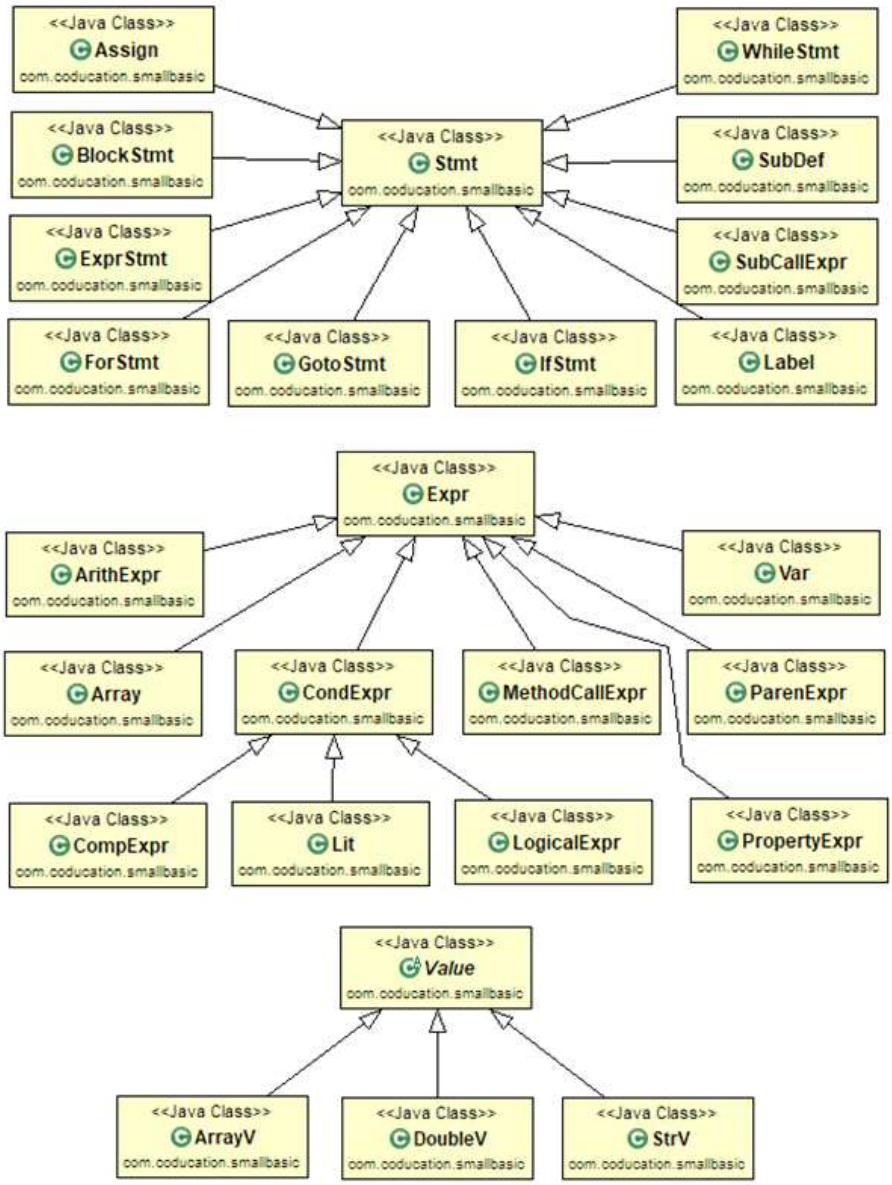


그림 6 스몰베이직 해석기 추상구문 트리

각 문장(Stmt), 표현식(Expr) 그리고 값(Value)가 가지고 있는 정보가 다르기 때문에, 추상구문 트리 클래스의 멤버변수에 대한 상세 내용을 정리한다. 먼저, 추상구문 트리 중 문장(Stmt)를 상속하는 클래스를 (표 5)에서 정리한다. 이 문장에는 할당문(Assign), 반복문 For(ForStmt), 분기문 Goto(GotoStmt), 조건문 If(IfStmt), 레이블(Label), 함수호출(SubCallExpr), 함수정의(SubDef), 반복문 While(WhileStmt)가 해당된다.

표 5 각 문장(Stmt)에 대한 멤버변수

클래스명	멤버변수 설명	
Assign	lhs(Expr)	: 할당문의 왼쪽에 오는 표현
	rhs(Expr)	: 할당문의 오른쪽에 오는 표현식
BlockStmt	stmts(Stmt[])	: 문장 리스트
ForStmt	var(Var)	: For문에서 사용되는 변수명
	init(Expr)	: 변수의 초기값
	end(Expr)	: For문이 종료될 변수값 조건
	step(Expr)	: 변수 증감식(기본값은 1)
	block(Stmt)	: For문을 통해 반복할 문장
GotoStmt	target(String)	: 분기할 레이블명
IfStmt	cond(Expr)	: If의 조건문 표현식
	_then(Expr)	: If의 조건문이 참일 때 실행할 문장
	_else(Expr)	: If의 조건문이 거짓일 때 실행할 문장
Label	label(String)	: 레이블 명
SubCallExpr	name(String)	: 호출하는 함수명
SubDef	name(String)	: 정의하는 함수명
	block(Stmt)	: 정의하는 함수에서 실행할 문장
WhileStmt	cond(Expr)	: While문이 실행될 조건문
	block(Stmt)	: While의 조건문이 참일 때 실행할 문장

다음의 (표 6)은 표현식(Expr)를 상속받아 확장한 클래스를 정리한다. (그림 6)에서 확인할 수 있듯이 산술식(ArithExpr), 배열(Array), 비교식(CompExpr), 상수(Lit), 논리식(LogicalExpr), 라이브러리 함수 호출(MethodCallExpr), 라이브러리 속성

(PropertyExpr), 변수(Var)가 이에 해당된다.

표 6 각 표현식(Expr)에 대한 멤버변수

클래스명	멤버변수 설명
ArithExpr	oprnd1(Expr) : 산술식의 첫 번째 피연산자
	op(int) : 산술식의 연산자(+, -, *, /)
	oprnd2(Expr) : 산술식의 두 번째 피연산자
Array	var(String) : 배열이름
	list(Expr[]) : 배열에서 사용된 index를 가지는 list
CompExpr	oprnd1(Expr) : 비교식의 첫 번째 피연산자
	op(int) : 비교식의 연산자(<, <=, >, >=, =, <>)
	oprnd2(Expr) : 비교식의 두 번째 피연산자
Lit	lit(String) : 상수값
	type(int) : 상수의 타입(NUM, STRING)
LogicalExpr	oprnd1(Expr) : 논리식의 첫 번째 피연산자
	op(int) : 논리식의 연산자(AND, OR)
	oprnd2(Expr) : 논리식의 두 번째 피연산자
MethodCallExpr	obj(String) : 라이브러리 명
	name(String) : 라이브러리 함수명
	args(Expr[]) : 라이브러리 함수에 전달될 인자리스트
PropertyExpr	obj(String) : 라이브러리 명
	name(String) : 라이브러리 속성명
Var	name(String) : 변수명

여기서 하나의 스몰베이직 프로그램을 통해 위에서 설명한 문장(Stmt)와 표현식(Expr)의 추상구문 트리로 변경된 결과를 확인해보자. (그림 7)은 9라인의 간단한 스몰베이직 프로그램이다. Mult10Times를 통해 2를 10번 곱하는 간단한 함수이며, 이 함수를 실행한 후 결과를 출력하는 것이다. 이 프로그램 전체는 총 9라인의 문장으로 BlockStmt로 묶어 표현해준다. 1번째 줄의 할당문, 2번째 줄의 함수호출문, 3번째 줄의 라이브러리 함수 호출문, 5번째 줄의 함수 정의문으로 크기가 4인 ArrayList를 가지는 BlockStmt가 만들어진다.

```
01: res = 10
02: Mult10Times()
03: TextWindow.WriteLine(res)
04:
05: Sub Mult10Times
06:   For i = 0 To 10
07:     res = res * 2
08:   EndFor
09: EndSub
```

그림 7 스몰베이직 프로그램 예제

```
new BlockStmt(
  [ new Assign(new Var("res"), new Lit(10))),
    new SubCallExpr("Mult10Times")),
    new MethodCallExpr(
       "TextWindow", "WriteLine", [new Var("res")]),
    new SubDef("Mult10Times",
      new BlockStmt(
       [ new ForStmt(new Var("i"), new Lit(0),
          new Lit(10), new Var(1),
          new BlockStmt(
            [ new Assign(new Var("res"),
              new ArithExpr(
                 new Var("res"), MULTIPLY, new Lit(2))) ])
       ])
  ])
```

그림 8 Java로 표현된 스몰베이직 프로그램 추상 구문 트리

(그림 8)은 추상 구문 트리가 만들어진 결과물이다. 먼저, 1번째 줄의 할당문에 대한 추상 구문 트리가 만들어지는 과정을 살펴보자. res라는 변수에 10이라는 상수를

할당하기 때문에, new Var(“res”)라는 변수와 new Lit(10)이라는 상수를 가지는 할당문이 만들어진다. 다음으로 2번째 줄의 함수호출 문장이 만들어지는 과정을 살펴보자. Mult10Times라는 이름의 함수를 호출하는 것이기 때문에, new SubCallExpr(“Mult10Times”)가 만들어진다. 다음으로 3번째 줄의 라이브러리 함수 호출 문장이 만들어지는 과정을 살펴보자. TextWindow라는 라이브러리 이름과 WriteLine이라는 함수명, res라는 변수 인자 한 개를 받는다. 각 함수마다 인자의 수는 다르기 때문에 ArrayList로 만들어주는데, 여기서는 크기가 1이고 new Var(“res”)를 가지는 리스트가 만들어진다. 즉, 이를 통해 new MethodCallExpr(“TextWindow”, “WriteLine”, [new Var(“res”)])를 만든다. 마지막으로 5번째 줄의 함수정의 문장을 만드는 과정이다. Mult10Times라는 이름을 가진 함수이며, 6번째 줄부터 8번째 줄까지가 함수가 호출되었을 때 실행해야 하는 문장으로 BlockStmt를 가지게 된다. 6번째 줄은 변수 i가 0부터 10까지 증가하며 7번째 줄의 문장을 반복 실행하는 반복문 For에 대한 추상 구문 트리를 만들어준다. (표 5)에서 ForStmt의 멤버변수에 맞춰, 변수인 new Var(“i”), 초기값인 new Lit(0), 종료값인 new Lit(10), 증감값 new Lit(1)을 가진다. 7번째 줄의 문장을 반복 실행하기 때문에 이에 대한 BlockStmt를 만들어준다. 7번째 줄은 res에 2를 곱하여 다시 res에 할당해주는 할당문이다. 먼저 res에 2를 곱하는 산술식인 new ArithExpr(new Var(“res”), MULTIPLY, new Lit(2)) 추상 구문 트리를 만든다. 이를 할당문에 추가하여 new Assign(new Var(“res”), new ArithExpr(...))을 만든다. 이 문장을 가지는 반복문 For는 new ForStmt (new Var(“i”), new Lit(0), new Lit(10), new Lit(1), new BlockStmt([new Assign(...)])으로 만들어진다. 이 문장을 가지는 함수 정의 문장은 new SubDef(“Mult10Times”, new BlockStmt([new ForStmt(...)])으로 만들어진다. 최종적으로 (그림 8)과 같은 추상 구문 트리가 만들어진다.

마지막으로, 값(Value)를 상속받아 확장하는 클래스에 대해 (표 7)에서 정리한다. 이 값들은 파싱단계에서 사용되는 것이 아닌 해석기의 계산(Eval) 단계에서 사용되어 만들어진다. 이 클래스는 배열(ArrayV), 정수/실수(DoubleV), 문자열(StrV)을 다룬다.

표 7 각 값(Value)에 대한 멤버변수

클래스명	멤버변수 설명
ArrayV	arrmap : 배열의 이름, 사용된 인덱스, 인덱스에 사용된 값
DoubleV	value : 정수/실수 상수값
StrV	v : 문자열 상수값

이 값에 대한 클래스는 계산(Eval) 단계 이후에 만들어지는 것이기 때문에 (그림 7)의 a 프로그램을 통해서 이 클래스가 만들어지는 과정을 설명한다. 7번째 줄의 “res * 2”는 res의 값에 2를 곱하는 것이기 때문에, 계산 단계가 시작되며 만들어진 환경(Env)에서 res의 값을 확인한다. 환경에 존재하는 res의 값이 10이라면, 10과 2를 곱한 값 20을 new DoubleV(20)으로 만들어 반환한다.

2.2 파싱 단계

개발한 스몰베이직 파서의 토큰 분석(token/lexical analysis)은 38개의 토큰을 받아들이는 유한 오토마타(finite automata)를 설계하여 구현하였고, 구문 분석(syntax analysis)은 59개의 생산 규칙(production rules)로 구성된 LALR(1) 문법을 작성하여 적용하였다.

스몰베이직 파싱 단계에 필요한 토큰 분석과 구문 분석에 관한 명세는 프로젝트 웹사이트에 공개하였다. 스몰베이직 파싱 단계에 필요한 이러한 명세 문서를 공개한 사례는 이전에 없었다. 다음의 (표 8)는 38개의 토큰 명세를 나타낸 것이다.

표 8 스몰베이직 해석기의 토큰 분석 명세

```
IF              ::= “If”
THEN            ::= “Then”
ELSEIF          ::= “ElseIf”
ELSE            ::= “Else”
ENDIF           ::= “EndIf”
WHILE           ::= “While”
ENDWHILE        ::= “EndWhile”
FOR             ::= “For”
TO              ::= “To”
STEP            ::= “Step”
ENDFOR          ::= “EndFor”
SUB             ::= “Sub”
ENDSUB          ::= “EndSub”
GOTO            ::= “Goto”
AND             ::= “And”
OR              ::= “Or”
OPEN_PARA       ::= “(”
CLOSE_PARA      ::= “)”
OPEN_BRACKET    ::= “[”
CLOSE_BRACKET   ::= “]”
DOT             ::= “.”
COMMA           ::= “,”
COLON           ::= “:”
ASSIGN          ::= “=”
LESS_THAN       ::= “<”
LESS_EQUAL      ::= “<=”
GREATER_THAN    ::= “>”
GREATER_EQUAL   ::= “>=”
NOT_EQUAL       ::= “<>”
PLUS            ::= “+”
MINUS           ::= “-”
MULTIPLY        ::= “*”
DIVIDE          ::= “/”
STR             ::= ”[^”]*”
NUM             ::= digit* | . digit* | digit+ . digit*
ID              ::= (letter | “_”) (letter | digit | “_”)*
CR              ::= “\n” | “’”
END_OF_TOKENS   ::= “$”
letter          ::= [a-zA-Z]
digit           ::= [0-9]
```

(표 9)는 스몰베이직의 구문 분석 명세로 총 59개의 생산규칙을 나열하였다.

표 9 스몰베이직 해석기의 구문 명세

```
Program              ::= MoreThanOneStmt
MoreThanOneStmt      ::= Stmt CR MoreThanOneStmt
                     | Stmt
Stmt                 ::= ExprStatement
                     | While Expr CRStmtCRs EndWhile
                     | ID :
                     | Goto ID
                     | For ID = Expr To Expr OptStep CRStmtCRs EndFor
                     | Sub ID CRStmtCRs EndSub
                     | If Expr Then CRStmtCRs MoreThanZeroElseIf
ExprStatement        ::= ID = Expr
                     | ID . ID = Expr
                     | ID . ID ( Exprs )
                     | ID ( )
                     | ID idxs = Expr
Exprs                ::= MoreThanOneExpr
                     | ε
MoreThanOneExpr      ::= Expr
                     | Expr , MoreThanOneExpr
Expr                 ::= CondExpr
CondExpr             ::= OrExpr
OrExpr               ::= OrExpr Or AndExpr
                     | AndExpr
AndExpr              ::= AndExpr And EqNeqExpr
                     | EqNeqExpr
EqNeqExpr            ::= EqNeqExpr = CompExpr
                     | EqNeqExpr <> CompExpr
                     | CompExpr
CompExpr             ::= CompExpr < AdditiveExpr
                     | CompExpr <= AdditiveExpr
                     | CompExpr > AdditiveExpr
                     | CompExpr >= AdditiveExpr
                     | AdditiveExpr
AdditiveExpr         ::= AdditiveExpr + MultiplicativeExpr
                     | AdditiveExpr - MultiplicativeExpr
                     | MultiplicativeExpr
MultiplicativeExpr   ::= MultiplicativeExpr * UnaryExpr
                     | MultiplicativeExpr / UnaryExpr
```

```
                    | UnaryExpr
UnaryExpr           ::= - Primary
                    | Primary
Primary             ::= NUM | STR | ( Expr ) | ID
                    | ID . ID | ID . ID ( Exprs ) | ID idxs
OptStep             ::= Step Expr
                    | ε
MoreThanZeroElseIf  ::= ElseIf Expr Then CRStmtCRs MoreThanZeroElseIf
                    | OptionalElse
OptionalElse        ::= Else CRStmtCRs EndIf
                    | EndIf
Idxs                ::= [ Expr ]
                    | [ Expr ] Idxs
CRStmtCRs           ::= CR TheRest
TheRest             ::= Stmt CR TheRest
                    | ε
```

스몰베이직 구문의 특성상 한 줄에 작성할 문장을 두 줄로 나누어 작성하면 구문 오류가 발생한다. 따라서 줄 바꿈 문자를 단지 토큰을 구분하는 역할(delimiter)로만 사용하는 것이 아니라 구문 분석에 필요한 핵심 토큰으로 사용한다.

구현한 파서를 테스트하기 위해 3장의 1절에서 설명한 호환성 검증을 위한 59개 스몰베이직 프로그램과 4개 벤치마크 프로그램을 사용하였다.

2.3 기본 블록 변환 단계

기본 블록 변환 단계에서 스몰베이직 프로그램의 제어흐름을 분석하여 기본 블록(basic block) 기반 제어 구조의 프로그램으로 변환하고 반복문 For와 While을 조건문 If, 라벨, 분기문 Goto로 대체한다. 반복문 For을 조건문 If로 변환하는 기본블록 생성 코드는 다음 (그림 9)와 같다. 이 외의 다른 문장을 기본블록으로 바꾸는 코드는 github의 MySmallBasic에 있다.[4)]

4) https://github.com/kwanghoon/MySmallBasic/blob/master/MySmallBasic/src/com/coducation/smallbasic/BBTransform.java

Input. For문장, 다음에 실행될 문장

Output. 레이블과 문장이 묶인 HashMap

Method. 1. 초기화 과정

1-1) For문의 변수 초기화 문장을 만들어준다.

1-2) For문의 변수 증감식에 대한 문장을 만들어준다.

1-3) For문의 조건식에 대한 문장을 만들어준다. 이 조건식은 아래의 두 조건 중 하나를 만족하는 논리식으로 작성된다.

- For문의 증가 값이 0보다 크고 변수값이 종료될 값보다 작거나 같다.

- For문의 증가 값이 0보다 작고 변수값이 종료될 값보다 크거나 같다.

2. For문의 문장(body)에 대해 transform 실행한다.

3. 위에서 만들어진 문장으로 If문을 만들어준다.

If-조건식: 3번 스텝을 통해 만들어진 조건문

If-참일 경우: For문의 body를 transform한 문장

If-거짓일 경우: 다음에 실행될 문장으로 넘어갈 Goto문

4. HashMap에 레이블과 기본블록을 넣어준다.

4-1) 3번 과정을 통해 만들어진 If문과 레이블 매핑

4-2) 다음에 실행될 문장과 레이블 매핑

5. 실행될 블록 문장에 다음 내용을 넣어 반환한다.

5-1) 1-1의 초기화 문장

5-2) 3번으로 분기할 Goto문

그림 9 For문에 대한 기본블록 변환 알고리즘

이 변환과정을 거치면 2장에 제시되었던 (그림 1)의 예제 프로그램은 (표 10)과 같은 프로그램이 된다.

표 10 기본 블록으로 변환된 프로그램

레이블	문장
$main	I = 1 Goto $L0

$L0	If 1 > 0 And I <= 20 Or 1 < 0 And I >= 20 Then
	  pic = Flickr.GetRandomPicture(“Korea”)
	  GraphicsWindow.DrawResizedImage(pic, 0, 0, 640, 480)
	  I = I + 1
	  Goto $L0
	Else
	  Goto $L1
	EndIf

$L1	

각 기본 블록은 다음과 같은 공통 특징을 가지고 있다.

- 기본 블록은 항상 라벨로 시작한다. 중간 문장에 라벨이 있어 기본 블록의 시작을 거치지 않고 점프해서 들어올 수 없으며, 중간 문장에 Goto문이 있어 끝까지 실행하지 않고 점프해서 나가는 경우는 없다.
- 기본 블록의 문장들 중 마지막에만 Goto문을 허용한다. 만일 Goto 문이 없으면 더 이상 실행할 문장이 없는 것으로 판단한다.
- 기본 블록은 특별한 확장 없이 스몰베이직 프로그램의 추상구문트리로 표현할 수 있다.

(표 10)에서 기본 블록으로 변환된 프로그램 구조를 보면 $main과 $L0 라벨로 시작하는 두 블록들은 각각 마지막 문장이 Goto문이다. $L0의 블록에서 If 조건이 참인 경우 마지막으로 실행될 문장이 Goto문이고, If 조건이 거짓인 경우 Goto문이 마지막 문장이고 이후 $L1의 블록에 실행할 문장이 없기 때문에 프로그램 실행을 종료한다.

2.4 실행 단계

실행 단계에서는 시작 라벨로 약속한 $main의 블록에서 프로그램 실행을 시작해

분기문으로 끝나는 블록들을 반복해서 실행한다. 그리고 마지막 문장이 더 이상 분기문이 아닌 블록을 실행하고 프로그램을 종료한다.

(표 10)과 같이 기본 블록들로 구조화된 프로그램을 실행하면 반복해서 분기하지만 다시 돌아오지 않는 패턴의 제어 흐름이 된다. 이 패턴의 제어 흐름을 구현하기 위해서 트램펄린(trampoline) 스타일[7]을 사용한다.

트램펄린 스타일은 아래의 드라이버 코드를 사용한다. 이 코드의 eval 함수는 변수 환경(env)과 실행할 블록을 인자로 받아 그 블록을 실행하고 다음 실행할 블록의 라벨을 반환한다. 이 라벨로 다음 실행할 블록을 찾아 다시 eval 함수로 반복 실행한다. 마치 트램펄린에서 점프하고 내려와서 다시 점프하는 과정을 반복하는 것과 유사하다.

```
currentLabel = “$main”;
env = initialEnvironment();
while (currentLabel != null) {
  Block block = blockMap.get(currentLabel);
  label = eval(env, block);
}
```

블록 단위의 실행을 위해 라벨 이름을 기본 블록으로 매핑하는 자료 구조(blockMap)는 변환 단계를 거친 블록들을 가지고 있고, 스몰베이직 프로그램은 전역 변수만을 허용하므로 변수 이름을 값으로 매핑하는 변수 환경(env)을 유지한다. 따라서 프로그램 시작 전에 변수 환경을 초기화하고 현재 선택된 (currentLabel이 가리키는) 블록을 실행하면서 이 변수 환경에 새로운 변수와 값의 쌍을 추가하거나 기존 변수의 값을 변경한다.

예를 들어, (표 10)의 예제에서 $main 라벨의 블록부터 실행한다. $main 블록의 마지막 문장 Goto $L0를 통해 다음에 $L0에 해당하는 블록을 실행한다. $L0의 If문에서 조건이 참이면 Goto $L0를 통해 다시 $L0 블록을 실행하고, 조건이 거짓이면 Goto $L1을 통해 $L1 블록을 실행한다(그림 10).

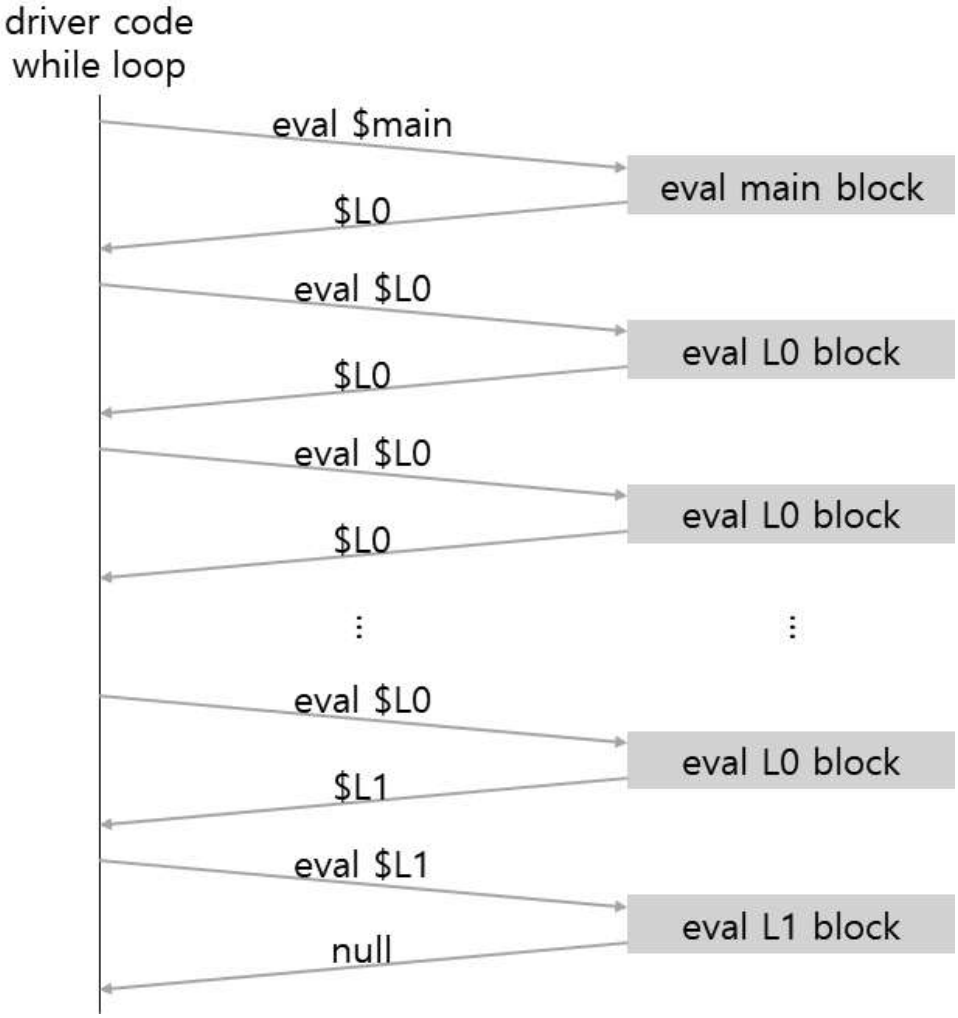


그림 10 변환된 기본 블록 계산방법

2.5 스몰베이직의 동적 형 변환

스몰베이직은 실행하면서 필요한 타입의 값으로 변환하여 사용하는 동적 형 변환을 이용한다. 스몰베이직 언어에서 문자열뿐만 아니라 정수와 실수를 포함한 숫자, 부울, 심지어는 배열도 모두 문자열로 변환할 수 있고, 그렇게 표현된 문자열을 다시 필요한 타입의 값으로 변환할 수 있다. 스몰베이직에서 다루는 동적 형 변환 예제를 살펴본 뒤, 동적 형 변환의 필요성을 알아보고 스몰베이직의 동적 형 변환을 정리한다.

2.5.1 동적 형 변환 예제

숫자, 부울, 배열에 대한 동적 형 변환 규칙을 예제를 통해 확인해 볼 수 있다.

● 문자열 타입의 값에서 숫자 타입의 값으로 변환

다음 (그림 11)은 문자열 타입의 값 "123"과 숫자 타입의 값 456을 곱하는 예제이다.

```
01: txt = "123"
02: num = txt * 456
03: TextWindow.WriteLine("Calc Result: " + num)
```

그림 11 동적 형 변환 규칙 예제: 숫자

C/C++/Java와 같은 언어에서는 이와 같은 수식이 허용되지 않는다. 그러나 스몰베이직에서는 동적 형 변환 규칙에 따라 이를 수행할 수 있다. 문자열 "123"이 숫자 123으로 변환된 뒤, 숫자 456과 곱해서 56088이라는 결과를 가진다. 숫자로 변환할 수 없는 문자열이 주어진 경우에는 이 값을 0으로 계산한다.

● 문자열 타입의 값에서 부울 타입의 값으로 변환

다음 (그림 12)는 문자열 타입의 값 "123"을 If의 조건문으로 사용하여 참일 경우 문자열 "txt is true"를, 거짓일 경우 "txt is false"를 출력하는 예제이다.

```
01: txt = "123"
02: If txt Then
03:   TextWindow.WriteLine("txt is true")
04: Else
05:   TextWindow.WriteLine("txt is false")
06: End If
```

그림 12 동적 형 변환 규칙 예제: 부울

(그림 12)의 예제의 결과로 문자열 "txt is false"가 콘솔창에 출력된다. 즉, txt가 "True"와 다르기 때문에 Else의 문장을 실행한다. txt가 문자열이 아닌 정수 123이여도 동일한 결과가 나온다. C/C++에서는 0을 False, 1을 True로 판단할 수 있지만, 스몰베이직에서는 "True"라는 문자열과 다르기 때문에 거짓으로 판단한다.

● 문자열 타입의 값에서 배열 타입의 값으로 변환

다음 (그림 13)은 문자열 타입의 값을 배열 타입의 값으로 변환하여 배열의 값의 출력하는 예제이다.

```
01: arr = “name=mysmallbasic;year=2019;txt=hello!;”
02:
03: TextWindow.WriteLine(“arr[name]: ” + arr[“name”])
04: TextWindow.WriteLine(“arr[age]: ” + arr[“age”])
05: TextWindow.WriteLine(“arr[txt]: ” + arr[“txt”])
```

그림 13 동적 형 변환 규칙 예제: 배열

(그림 13) 같이 arr에 “name=mysmallbasic;year=2019;txt=hello;”를 대입하면, 이 문자열은 세미콜론(;)을 기준으로 배열의 인덱스와 값의 쌍을 나눈다. =을 통해서 배열 인덱스와 값으로 나눠 저장한다. 즉, 이 배열 원소 arr[“name”]은 문자열 “mysmallbasic”을 가진다. 동일하게 arr[“year”]는 문자열 “2019”를, arr[“txt”]는 문자열 “hello”를 가지고 있다. 이처럼 스몰베이직은 문자열에서 필요에 따라 동적으로 정수/실수, 부울, 배열로 변환 될 수 있으며, 역으로 정수/실수, 부울, 배열에서 문자열로 변환될 수 있다.

2.5.2 스몰베이직 동적 형 변환 규칙

이러한 동적 형 변환 특성을 명확히 이해하는 것은 스몰베이직 해석기를 구현할 때 매우 중요하지만 이에 대해 설명하는 문서가 없다. 이 절에서 마이크로소프트 코딩 환경을 역공학 분석한 스몰베이직의 동적 형 변환을 정리한다.

스몰베이직에서 숫자, 부울, 문자열, 배열을 어떻게 작성하고, 각 타입의 값을 문자열로 변환하는 방법을 예를 들어 설명한다. 숫자는 123 또는 123.456로 작성하면 내부적으로 “123” 또는 “123.456”과 같이 숫자를 담고 있는 문자열로 변환한다. 부울 값은 “True”와 “False”와 같이 대소문자 구분하지 않는 문자열로 작성한다.

일차원 배열은 키(첨자)와 값으로 표현한다. 앞에서 보인 예와 같이 첨자 “name”, “year”, “txt”의 배열 원소가 “mysmallbasic”, “2019”, “”hello“이면, 이 배열은 키와 값을 등호로 구분하고 각 배열 원소를 세미콜론으로 구분하는 형식의 문자열 “name=mysmallbasic;year=2019;txt=hello”로 정의된다. 다차원 배열은 키(첨자)와

한 차원 낮은 배열의 (문자열로 변환된) 값을 매핑한 것으로 일반화하여 표현할 수 있다.

배열 첨자는 대소문자를 구분하지 않는 문자열로 정의한다. 따라서 숫자뿐만 아니라 문자열도 배열의 첨자로 사용할 수 있고, 심지어 문자열로 표현할 수 있는 배열도 첨자로 사용할 수 있다.

참고로 모든 변수와 배열의 원소는 빈 문자열 “”로 초기화되도록 정의되어 있다. 따라서 자바에서와 같이 널(Null) 값을 허용하지 않는다.

스몰베이직에서 타입 간 변환 방법을 설명하기 위해 프로그램으로 작성할 수 있는 4가지 타입(T)을 number, string, boolean, array(T)라 하자. 수직으로 배치된 타입을 가로로 배치된 타입으로 변환하는 방법을 (표 11)을 통해 설명한다.

표 11 스몰베이직 타입 변환 방법

=>	number	string	boolean	array
number		(1)	(2)	(3)
string	(4)		(5)	(6)
boolean	(7)	(8)		(9)
array	(10)	(11)	(12)	

(1) 123 또는 123.456을 “123” 또는 “123.456”으로 변환

(2) 모든 숫자를 거짓 “False”로 변환

(3) 모든 숫자를 원소가 없는 배열로 변환

(4) 숫자를 담고 있는 문자열 “123” 또는 “123.456”을 123 또는 123.456으로 변환. 그 외의 모든 문자열은 0으로 변환 (예: 빈 문자열 “”, “abc”)

(5) 대소문자를 구분하지 않고 문자열을 비교할 때 “True”와 동일하면 참, 아니면 거짓으로 변환

(6) 앞서 설명한 형식을 따르는 문자열을 키(첨자)와 값의 매핑인 배열로 변환. 이와 관련하여 아래에 제시할 보충 설명을 참고한다.

(7) 참과 거짓 모두 0으로 변환

(8) 참은 “True”와 거짓은 “False”

(9) 참과 거짓 모두 원소가 없는 배열로 변환
(10) 모든 배열은 0으로 변환
(11) 앞서 설명한 형식에 따라 배열을 문자열로 변환
(12) 모든 배열을 거짓으로 변환

앞에서 설명한 스몰베이직의 4가지 타입 간 변환 방법 은 문맥에 독립적으로 정의한 것이다. 연산자에 따라 이 정의만으로 충분히 설명할 수 없는 타입 변환이 발생할 수 있다. 예를 들어, 1 + “True”에서 “True”를 숫자 타입으로 변환하면 1+0이 되고 문자열 타입으로 변환하면 “1True”가 된다. 마이크로소프트 스몰베이직은 이 경우 + 연산자를 숫자 더하기로 해석하기보다 문자열 붙이기로 해석한다. 마이스몰베이직도 이를 그대로 따르도록 해석기를 구현하였다. 만일 “True”를 숫자 0으로 변환하는 방법을 먼저 적용한다면 1 + 0이 되어 잘못된 결과를 낼 것이다. 이러한 차이가 발생할 가능성이 있기 때문에 동적 형 변환에 대한 정확한 정의가 필요하다.

반면에 1 - “True”의 경우 부울값 “True”를 숫자로 변환하기 때문에 앞에서 설명한 방법에 따라 1 – 0으로 변환한다. 마이너스 연산자뿐만 아니라 곱하기, 나누기 연산자도 동일한 변환 방식을 사용한다.

1 + arr과 같이 부울값이 아닌 배열의 경우도 비슷하다. + 연산자를 문자열 붙이기로 우선 해석하여 배열을 문자열로 변환하고 (그리고 숫자 1도 문자열로 바꾼 다음) 연산자를 적용한다. 마찬가지로 1 - arr의 경우 1 – 0으로 바꾸어 진행한다.

크기를 비교하는 연산자(<, >, <=, >=)는 숫자에 대해서만 정의되어 있다. 따라서 “abc” < “abcdef”를 사전식 순서로 생각하면 참이 되지만 스몰베이직에서 양쪽 문자열을 숫자로 바꾸면 모두 0이 되므로 0 < 0은 거짓이다.

같다를 비교하는 연산자(=)는 문자열로 변환한 다음 이 문자열이 같으면 참이라고 판단하도록 정의되어 있다.

마지막으로 배열을 문자열로 변환하는 과정에서 아주 미묘한 상황이 발생할 수 있음을 설명한다. 앞서 사용한 배열을 표현한 올바른 문자열 예는 “0=123;1=456;2=789”이다. 만일 어떤 이유에서 세 번째 등호가 빠진 “0=123;1=456;2789” 문자열을 배열 변수 arr에 대입하면 그 결과는 무엇일까? 형식 오류가 발생했기 때문에 아무 원소도 없는 빈 배열이 될 수도 있고, 또는 형식이 올바른 첨자 0과 1의 원소만 포함하고 그 이후 “2789”를 무시한 원소가 2개 있는 배열

이 될 수도 있다. 마이크로소프트 스몰베이직에서 후자의 방법으로 정의하고 있음을 확인하였다. 이렇게 사소하지만 미묘한 타입 변환의 차이가 프로그램 실행 결과에 영향을 준다. 따라서 명확한 정의가 있어야 서로 다른 스몰베이직 코딩 환경 간 실행 결과가 동일하도록 보장할 수 있다.

제 3 절 그래픽스 아키텍처 설계 및 구현

오픈소스 소프트웨어 프로젝트 마이스몰베이직은 교육용 프로그래밍 언어로써 다양한 라이브러리를 제공한다. 그 중 입문자의 흥미를 가장 높이는 것은 눈으로 직접 확인할 수 있는 그래픽스 관련 라이브러리이다. 이 절에서는 스몰베이직 문서에서 소개되지 않았던 그래픽스 아키텍처 라이브러리와 자료구조, 레이어 구조에 대해서 상세하게 설명한다.

3.1 그래픽스 아키텍처

스몰베이직에서 제공하는 라이브러리 중 그래픽스 라이브러리의 기반이 되는 GraphicsWindow가 있다. 또 이 그래픽스 라이브러리와 관련되어 스몰베이직에서 제공하는 표준 라이브러리 Shapes, Controls가 있다. 총 3개의 라이브러리가 하나의 그래픽스 창에 표현되어야 하기 때문에 레이어 개념을 이용한다. 다음의 (그림 14)는 스몰베이직에서 제공하는 표준라이브러리를 이용한 그래픽스 프로그램이다.

```
01: shape1 = Shapes.AddEllipse(70, 70)
02: Shapes.Move(shape1, 20, 20)
03:
04: control1 = Controls.AddButton(“hello”, 40, 20)
05:
06: GraphicsWindow.BrushColor = "Gold"
07: GraphicsWindow.FillRectangle(0, 0, 200, 100)
```

그림 14 그래픽스 라이브러리 레이어 계층 예제

이 프로그램을 스몰베이직에서 실행하면 (그림 15)와 같은 실행 결과를 보여준다.

(그림 14)는 Shapes, Controls, GraphicsWindow 순서대로 작성되었음에도 불구하고, GraphicsWindow로 그린 사각형이 가장 아래에 그려져 있음을 확인할 수 있다. 이를 통해 Shapes와 Controls는 GraphicsWindow가 그려지는 레이어 위에 그려짐을 확인할 수 있다.

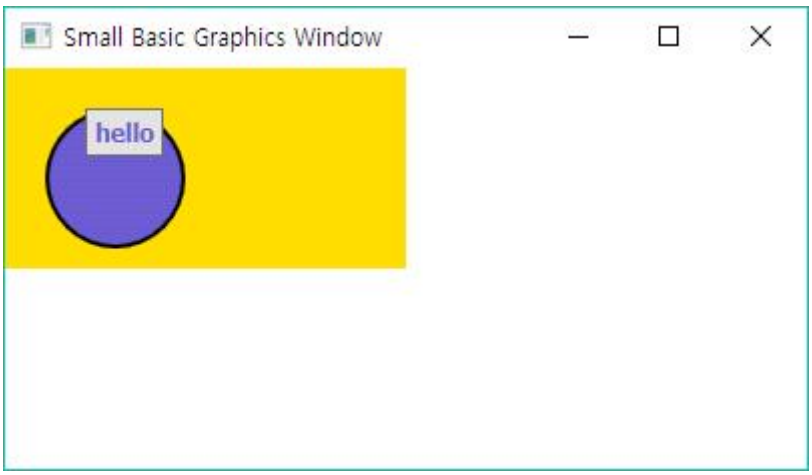


그림 15 그래픽스 라이브러리 프로그램 실행화면

그렇다면 Shapes와 Controls 라이브러리는 같은 레이어 상에 그려질지 확인해 볼 필요가 있다. (그림 14)의 프로그램을 그대로 사용하여 4번째 라인에 작성된 코드를 1번째 라인으로 끌어올려 실행했을 때 다음 (그림 16)과 같은 결과를 확인할 수 있다.

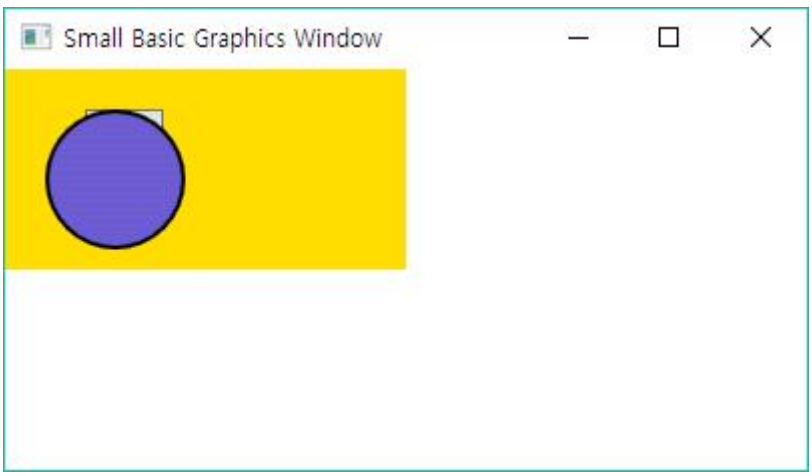


그림 16 레이어 확인을 위한 프로그램 실행화면

(그림 15)와는 다르게 Shapes를 통해 그려진 원이 Controls를 통해 그려진 버튼 위에 그려져 있음을 확인할 수 있다. 이를 통해 Shapes와 Controls는 동일한 레이어 계층에 존재하며, 작성되는 순서에 따라 그려짐을 확인할 수 있다. 이러한 그래픽스 구조 분석을 통해, 마이스몰베이직은 첫 번째 레이어에 GraphicsWindow에 대한 도형을 그리며, 두 번째 레이어에 Shapes, Controls에 대한 내용을 보여준다.

마이스몰베이직은 오픈소스 소프트웨어로 라이브러리의 확장이 가능하다. 이 점을 이용해 초보자를 위한 확장 라이브러리를 개발하였다. 그 중 그래픽스 창에 보여지는 Chart, Graph, Video 라이브러리가 있는데, 이는 스몰베이직에서 제공하는 라이브러리가 아니기 때문에 세 번째 레이어 개념을 추가하였다. (그림 17)은 마이스몰베이직의 그래픽스 라이브러리를 이용하여 그려지는 레이어 계층을 보여준다.

```
01: myVideo = Video.Play(“some url”)
02: Video.SetLocation(myVideo, 80, 70)
03: Video.SetSize(myVideo, 480, 480)
04:
05: shape1 = Shapes.AddEllipse(70, 70)
06: Shapes.Move(shape1, 20, 20)
07: GraphicsWindow.BrushColor = "Gold"
08: GraphicsWindow.FillRectangle(0, 0, 200, 100)
```

그림 17 그래픽 관련 라이브러리 레이어 계층 예제

총 8라인의 예제 프로그램은 순서대로 Video를 통해 동영상을, Shapes를 통해 타원을, GraphicsWindow를 통해 사각형을 그린다. 이 프로그램의 실행결과 및 마이스몰베이직 그래픽스 라이브러리 레이어 계층도는 (그림 18)과 같다. 가장 아래에 있는 첫 번째 레이어는 GraphicsWindow 라이브러리의 도형을 그리고, 두 번째 레이어는 Shapes와 Controls를 그려줌을 확인할 수 있다. 또 세 번째 레이어는 스몰베이직에 존재하지 않지만 마이스몰베이직에서 확장 개발한 라이브러리를 보여주는 계층으로 정의하였다.

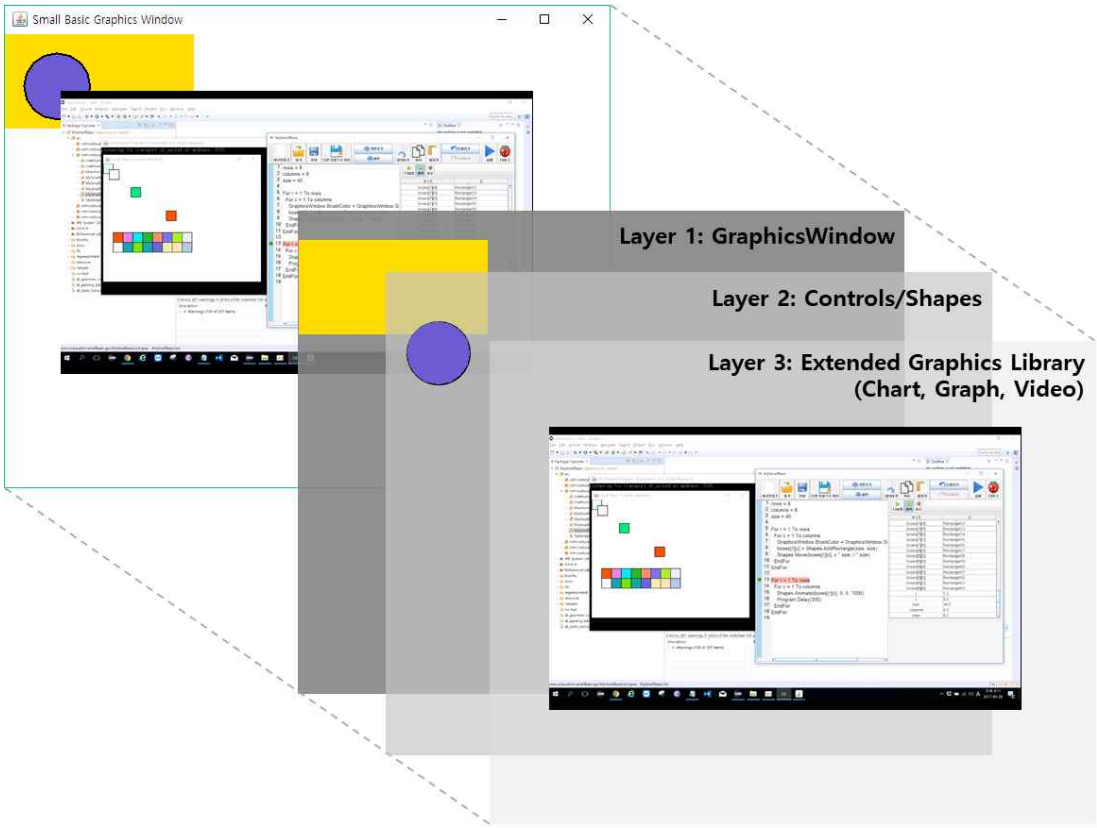


그림 18 그래픽 라이브러리 레이어 계층도

3.2 그래픽스 라이브러리

이 절에서는 스몰베이직에서 제공하는 표준 라이브러리 중 그래픽스 라이브러리인 GraphicsWindow, Shapes, Controls를 설명하며, 각각의 특징에 대해 설명한다.

3.2.1 GraphicsWindow

가장 기본이 되는 그래픽스 라이브러리는 GraphicsWindow이다. 이 라이브러리는 그래픽스 창을 하나 띄워주며, 이 위에 사용자는 점, 선, 타원, 사각형 등의 도형을 손쉽게 그릴 수 있다. GraphicsWindow는 Java의 JFrame을 이용하여 화면을 보여준다. 그리고 도형을 그리거나 텍스트를 쓰기 위해서 JPanel을 이용한다. GraphicsWindow에서 제공하는 함수는 다음 (표 12)과 같다.

표 12 GraphicsWindow 라이브러리 함수

함수명	설명
Clear	그래픽 창에 표시된 모든 것을 지움
DrawBoundText	지정한 위치에 지정한 길이 범위 안에서 글자 표시
DrawEllipse	지정한 위치에 타원을 그림
DrawImage	지정한 위치에 그림을 표시
DrawLine	한 지점에서 다른 지점으로 직선을 그림
DrawRectangle	지정한 위치에 사각형을 그림
DrawResizedImage	지정한 위치에 지정한 크기로 그림을 표시
DrawText	지정한 위치에 글자 표시
DrawTriangle	지정한 위치에 삼각형을 그림
FillEllipse	지정한 위치에 색이 채워진 타원을 그림
FillRectangle	지정한 위치에 색이 채워진 사각형을 그림
FillTriangle	지정한 위치에 색이 채워진 삼각형을 그림
GetColorFromRGB	주어진 RGB로 색을 만들어 반환
GetPixel	지정한 위치의 픽셀 색상값 반환
GetRandomColor	표현 가능한 임의의 색 반환
Hide	그래픽 창 숨김
SetPixel	지정한 위치에 지정한 색으로 점을 찍음
Show	그래픽 창 표시
ShowMessage	메시지 상자 표시

GraphicsWindow에서는 키보드와 관련된 이벤트, 마우스와 관련된 이벤트를 다루고 있다. 키보드와 관련된 이벤트 함수를 받는 KeyUp, KeyDown, TextInput이 있으며, 마우스와 관련된 이벤트 함수를 받는 MouseUp, MouseDown, MouseMove가 있다. GraphicsWindow에서 제공하는 변수는 총 18개로 다음의 (표 13)과 같다.

표 13 GraphicsWindow 라이브러리 변수

변수명	설명
BackgroundColor	그래픽 창의 배경색을 설정하거나 가져옴
BrushColor	그래픽 창에 도형을 칠할 때 사용되는 색상
CanResize	그래픽 창의 사이즈를 마우스로 조절할 수 있는지 결정
FontBold	글씨를 굵게할 것인지에 대한 결정
FontItalic	글씨를 기울일 것인지에 대한 결정
FontName	사용될 폰트의 이름
FontSize	글씨의 크기
Height	그래픽 창의 높이
LastKey	그래픽 창에서 마지막으로 누르거나 놓은 키
LastText	그래픽 창에서 마지막으로 입력된 글자
Left	그래픽 창의 왼쪽 위치
MouseX	그래픽 창에 비례한 마우스의 x축 위치
MouseY	그래픽 창에 비례한 마우스의 y축 위치
PenColor	그래픽 창에 도형을 그릴 때 사용되는 색상
PenWidth	그래픽 창에 도형을 그릴 때 사용되는 펜의 두께
Title	그래픽 창의 제목
Top	그래픽 창의 위쪽 위치
Width	그래픽 창의 너비

GraphicsWindow를 통해 그릴 수 있는 도형을 관리하기 위해 필요한 다양한 정보를 가지고 있는 Cmd라는 추상클래스를 사용하며, 이를 순차적으로 그려주기 위해 Java에서 제공하는 자료구조 ArrayList를 사용한다. 추상클래스 Cmd는 다음과 같이 구성되어 있다.

```
private static abstract class Cmd {
  int layer;        // layer가 그려질 번호
  boolean show;     // 도형이 보여질지에 대한 값
  int cmd;          // draw/fill 선, 원, 삼각형, 사각형, 텍스트
  double x, y;      // x, y 좌표
  float opacity;    // 투명도
  double scaleX, scaleY; // 확대/축소 값
  double degree;          // 회전 각도
}
```

라이브러리를 통해 그릴 수 있는 도형들은 기본적으로 위와 같은 속성을 가진다. 추가적으로 필요한 속성들은 이 추상클래스를 확장하여 필요한 추가정보를 저장한다. 예를 들어, 선을 그리는 경우에는 시작점 (x1, y1)과 도착점 (x2, y2)의 정보를 저장해야한다. 또 사각형을 그리는 경우에는 사각형의 너비(width)와 높이(height)를 저장해야 한다. Cmd를 확장하여 다음과 같은 클래스로 작성할 수 있다.

```
private static class DrawLine extends Cmd {
  int x1, y1;       // Start Point
  int x2, y2;       // End Point
}
...
private static class DrawRectangle extends Cmd {
  int width, height;      // Rectangle Width and Height
}
```

이와 같은 변수를 이용하여 GraphicsWindow에서 paintComponent를 통해 도형을 그려준다. paintComponent는 기본적으로 Graphics를 이용하지만, 선의 두께 등의 다양한 옵션을 설정해주기 위해서는 Graphics2D를 사용해야 한다. GraphicsWindow에서 구현한 paintComponent는 다음 (그림 19)와 같다.

```
public void paintComponent(Graphics g) {
  ...
  Graphics2D g2 = (Graphics2D) g;
  ...
  for (int layer = 1; layer <= 2; layer++) {
    ...
    if (cmd.getLayer() != layer) continue;
      case DRAWLINE:
        ...
        g2.setComposite(AlphaComposite.getInstance
                    AlphaComposite.SRC_OVER, (float) opacity);
        g2.rotate(java.lang.Math.toRadians(degree),
              ((x1 + x2) / 2) * zoomX, ((y1 + y2) / 2) * zoomY);
        g2.scale(scaleX, scaleY);
        g2.setStroke(new BasicStroke((float) penwidth);
        g2.setColor(new Color(color));
        g2.drawLine(x1, y1, x2, y2);
        g2.rotate(java.lang.Math.toRadians(-degree),
                                (x1 + x2) / 2, (y1 + y2) / 2);
  ...
  }
}
```

그림 19 GraphicsWindow의 paintComponent 함수

3.2.2 Shapes

GraphicwWindow에서 제공하는 다양한 도형을 그릴 수 있는 Shapes 라이브러리가 있다. Shapes는 GraphicsWindow와 다르게 도형의 선과 색을 칠하는 것이 하나로 묶여있다는 차이점을 가진다. 또한 도형을 동적으로 움직이게 할 수 있는 기능을 제공하고 있다. Shapes 라이브러리에서 제공하는 함수는 다음의 (표 14)와 같다.

표 14 Shapes 라이브러리 함수

함수명	설명
AddRectangle	그래픽 창에 사각형을 그림
AddEllipse	그래픽 창에 타원을 그림
AddTriangle	그래픽 창에 삼각형을 그림
AddLine	그래픽 창에 선을 그림
AddImage	그래픽 창에 이미지를 그림
AddText	그래픽 창에 글씨 작성
SetText	텍스트 도형의 글씨를 설정함
Remove	도형 삭제
Move	도형을 지정한 위치로 옮김
Rotate	도형 회전
Zoom	도형 확대/축소
Animate	도형을 지정한 위치까지 움직임
GetLeft	그래픽스 창을 기준으로 도형의 왼쪽 여백값 반환
GetTop	그래픽스 창을 기준으로 도형의 위쪽 여백값 반환
GetOpacity	도형의 투명도 반환
SetOpacity	도형의 투명도 설정
HideShape	도형 숨김
ShowShape	도형 표시

GraphicsWindow와 동일한 도형을 그리기 때문에, Shapes 또한 Cmd를 통해 자신이 그려질 도형을 관리한다. Draw와 Fill이 하나로 묶여야 하기 때문에 ArrayList를 이용하며, 이를 묶어 그려주는 GraphicsWindow는 HashMap을 이용해 도형의 이름과 도형 리스트를 묶어 관리한다. 또한 GraphicsWindow와는 다르게 Shapes는 도형을 그리고 그에 맞는 이름을 반환하는데, `AddRectangle`을 한 경우 `"Rectangle1"`이라는 도형의 이름을 반환해준다. 동일하게 `AddTriangle`은 `"Triangle1"`을 반환하며, 한 번 더 `AddTriangle`을 할 경우 `"Triangle2"`를 반환한다.

3.2.3 Controls

사용자와 상호작용할 수 있는 버튼, 텍스트필드를 제공하는 Controls 라이브러리가 있다. 이 Controls 라이브러리에서 제공하는 함수는 다음의 (표 15)와 같다.

표 15 Controls 라이브러리 함수

함수명	설명
AddButton	그래픽 창에 버튼 추가
GetButtonCaption	버튼에 작성된 텍스트 반환
SetButtonCaption	버튼에 작성될 텍스트 설정
AddTextBox	그래픽 창에 텍스트 필드 추가
AddMultiLineTextBox	그래픽 창에 TextArea 추가
GetTextBoxText	텍스트 필드에 작성된 텍스트 반환
SetTextBoxText	텍스트 필드에 작성될 텍스트 설정
Remove	버튼/텍스트 필드 삭제
Move	버튼/텍스트 필드 이동
SetSize	버튼/텍스트 필드 크기 설정
HideControl	버튼/텍스트 필드 숨김
ShowControl	버튼/텍스트 필드 표시

Controls는 사용자와의 상호작용을 제공하기 때문에, 버튼을 클릭했을 때의 이벤트, 텍스트 필드에 타이핑을 했을 때의 이벤트를 제공한다. 그리고 이 이벤트에 사용될 `LastClickedButton`과 `LastTypedTextBox` 변수를 제공한다. Java에서는 각 버튼이 클릭되었을 때의 이벤트를 설정해주기 위해서는 이벤트 리스너를 모두 연결해 주어야 한다. 그러나 스몰베이직에서는 버튼에 대한 이벤트를 설정하기 위해 이벤트 리스너를 하나씩 만들어줄 필요가 없다는 것이 큰 특징이다. (그림 20)은 스몰베이직에서 버튼 클릭에 대한 이벤트를 만들어 주는 예제이다.

```
01:  first = Controls.AddButton("first button", 10, 50)
02:  second = Controls.AddButton("second button", 110, 50)
03:  third = Controls.AddButton("third button", 230, 50)
04:
05:  Controls.ButtonClicked = displayPopUp
06:
07:  Sub displayPopUp
08:    clickedButton = Controls.LastClickedButton
09:    txt = Controls.GetButtonCaption(clickedButton)
10:
11:    GraphicsWindow.ShowMessage("Clicked " + txt, "Event")
12:  EndSub
```

그림 20 Controls 라이브러리 이벤트 연결 예제

(그림 20)의 프로그램을 실행해서 버튼을 클릭하면 (그림 21)과 같은 결과 화면을 보여준다. 7번째 라인의 displayPopUp 함수를 살펴보면, Controls 라이브러리를 통해 마지막으로 클릭된 버튼을 찾아내고, 그 버튼의 텍스트를 가져와 팝업메시지 창에 띄워준다.

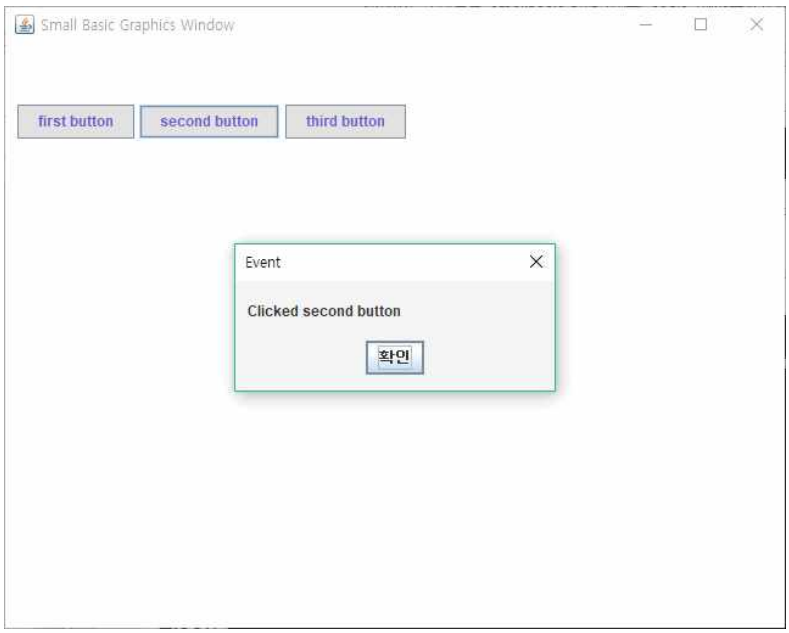


그림 21 Controls 라이브러리 이벤트 실행화면

3.3 그래픽스 아키텍처: 애니메이션 기능

그래픽스는 교육의 재미를 위해 쉽게 코딩할 수 있는 다양한 함수를 가지고 있다. 그 중 그래픽스와 관련된 라이브러리 Shapes에서는 Animate라는 함수를 지원한다. 아래의 (그림 22)는 애니메이션 기능을 이용하는 예제로 (그림 23)의 실행화면을 보여준다.

```
... Set boxes ...

For r = 1 To rows
  For c = 1 To columns
    Shapes.Animate(boxes[r][c], 0, 0, 1000)
    Program.Delay(300)
  EndFor
EndFor
```

그림 22 애니메이션 기능 예제

이 Animate라는 함수는 주어진 시간동안 주어진 위치까지 도형이 이동한다. 즉, (그림 22)에서 사용된 Animate 함수는 1초 동안 boxes[r][c]를 (0, 0)으로 이동한다.

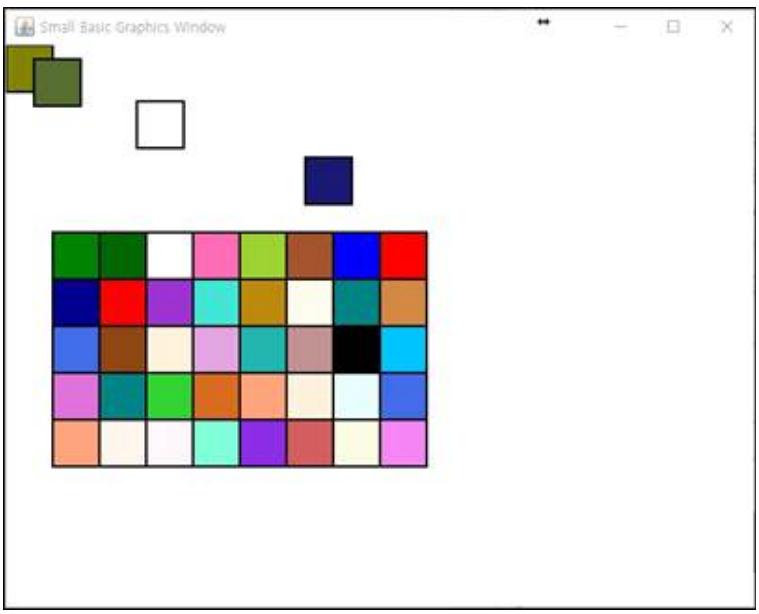


그림 23 애니메이션 기능 예제: 실행화면

Animate 함수는 주어진 시간이 0이라면 이 도형은 바로 주어진 위치로 이동한다. 주어진 시간이 0이 아니라면 이 도형은 ActionListener를 통해 주어진 위치까지 나누어 이동하게 된다. 이 Animate를 구현하기 위해 자바에서 제공하는 Timer를 이용하여 ActionListener를 연결해 주었다. Animate 함수는 총 4개의 인자를 받는다. 첫 번째는 이동할 도형의 이름, 두 번째는 이동할 위치의 x좌표, 세 번째는 이동할 위치의 y좌표, 네 번째는 애니메이션을 실행할 milliseconds 단위의 값을 받는다.

```
class Shapes {
...
  // arg0: 애니메이션을 실행할 Shapes 이름
  // arg1(x), args2(y): 이동할 x, y 좌표
  // arg3(duration): 애니메이션을 실행할 milliseconds
  static Animate(ArrayList<Value> args) {
    // cmds = HashMap에서 Shapes 이름을 통해 도형 리스트 찾기
    ...
    MyActionListener animate_action
        = new MyActionListener(cmds, x, y, duration);
    javax.swing.Timer timer = new Timer(100, animate_action);
    ...
  }
}
```

Timer에서 이벤트가 발생하면 ActionListener의 actionPerformed를 통해 도형이 이동할 위치가 정해지고 repaint를 호출한다. Animate와 ActionListener는 다음 (그림 24)와 같이 구현되어 있다. 위의 코드에서 MyActionListener 객체를 생성하며, 이동할 도형의 리스트, 좌표, 이벤트가 발생할 시간을 넘겨준다.

```
static class MyActionListener implements ActionListener {
  int i = 1;                    // 현재 반복된 횟수
  ArrayList<Cmd> cmds;          // 움직일 도형리스트
  int times;                    // 반복할 횟수
  double x, y;                  // x, y 좌표
  public javax.swing.Timer timer;

  public void actionPerformed(ActionEvent e) {
    int n = 0;
    for (Cmd cmd: cmds) {
       ...
       x = cmd.x + cmd.x / times;
       y = cmd.y + cmd.y / times;
       cmd.Move(x, y);
       ...
    }
    panel.repaint();
    ...
    if (i > times)
      timer.stop();
  }
}
```

그림 24 Animate 함수 구현

Animate 함수를 통해 도형을 이동할 때 기존의 위치에서 이동할 위치까지를 횟수로 나누어 조금씩 이동한다. 이 반복할 횟수를 크게 설정해주면 Animate 함수가 자연스럽게 넘어가지만, 오버헤드가 발생한다는 문제점이 있다. 반대로 반복할 횟수를 작게 설정해주면 Animate 함수가 부자연스럽게 이동하지만, 오버헤드 문제가 없어 프로그램이 느려지는 현상이 없다.

제 4 절 라이브러리 아키텍쳐

마이스몰베이직의 오픈소스 프로젝트에 외부 개발자가 참여하여 쉽게 기여할 수 있는지 확인할 목적으로 확장 라이브러리를 시험 개발하였다. 커뮤니티에서 다양한 확장 라이브러리를 개발하면 학생들의 흥미를 높이는 코딩 교육을 기획하고 진행하는데 도움이 될 것이다. 기존 스몰베이직 표준 라이브러리에 10가지 새로운 라이브러리 Assert, Chart, Database, Facebook, Graph, Hamster, List, Tree, Video, Weka 를 추가로 확장 개발하여 목표하는 바를 검증하였다.

외부 개발자가 이 프로젝트에 참여하여 새로운 라이브러리를 개발하고 추가하더라도 이미 개발된 스몰베이직 해석기와 코딩 환경을 변경하지 않도록 독립적인 라이브러리 구조를 설계할 필요가 있다.

기본적인 구조는 다음과 같이 요약할 수 있다. 각 사항에 대해 이후 자세히 설명한다.

- 스몰베이직의 각 라이브러리는 자바 클래스이다.
- 라이브러리 함수는 이 클래스의 멤버함수로, 라이브러리 변수는 이 클래스의 멤버변수이다.

기본적으로 새로운 라이브러리는 동일한 이름의 자바 클래스로 작성하고 라이브러리 클래스들을 모아놓은 자바 패키지(com.coducation.smallbasic.lib)에 포함시킨다. 스몰베이직 프로그램에서 이 라이브러리의 함수와 변수를 이 클래스의 정적 메소드와 정적 멤버 변수로 구현하다. 이러한 방식으로 라이브러리를 위한 자바 클래스를 만들면 자바 리플렉션(Reflection) 방법으로 라이브러리 함수와 변수를 접근하도록 구현할 수 있다. 예를 들어, 스몰베이직 해석기에서 (그림 1)의 `GraphicsWindow.DrawResizedImage` 함수 호출을 구현하는 방법은 문자열 **"GraphicsWindow"**와 **"DrawResizedImage"**를 자바 리플렉션 방법으로 해당 클래스와 메소드를 각각 찾아 호출한다.

4.1 확장 라이브러리 함수 작성

먼저 (그림 25)는 라이브러리 함수를 작성하는 방법이다. 기본적으로 스몰베이직 라

이브러리 함수의 인자 수는 가변적일 수 있기 때문에 모든 라이브러리 함수의 하나의 인자 또는 여러 개의 인자들을 Java의 ArrayList<Value> 타입의 객체 하나를 전달하여 표현한다. 라이브러리 함수의 인자가 없을 때도 이 ArrayList<Value> 객체를 만들어 넘겨야 한다. 이 때 Value는 스몰베이직 값을 Java로 표현하는 클래스들 StrV, DoubleV, ArrayV의 기반 클래스이다. 라이브러리 함수의 리턴 값이 있다면 Value 클래스를 라이브러리 함수를 구현하는 메소드의 리턴타입으로 지정한다.

```
package com.coducation.smallbasic.lib;

public class GraphicsWindow {
  ...
  public static void DrawResizedImage(ArrayList<Value> args) {
    if (args.size() == 5) {
      Value imgNameV = args.get(0);
      ...
      String imgNmae = ((StrV) imgNameV).getValue();
      ...
      panel.DrawResizedImage(imgName, x, y, width, height);
    }
    else
      throw new InterpretException(“DrawResizedImage:” +
                    “Unexpected # of args: ” + args.size());
  }
}
```

그림 25 확장 라이브러리 함수 작성 방법

라이브러리 함수를 구현하는 메소드에서 처음으로 할 일은 ArrayList<Value> 객체 리스트에 포함된 인자 개수를 카운팅하는 것이다. GraphicsWindow의 DrawResizedImage는 인자 5개가 있는지 확인한다. 인자의 개수가 예상하는 개수가 다르다면 InterpretException 예외를 발생시킨다. 인자의 개수가 기대한 것과 일치한다면 각 인자의 타입을 조사하고 실제 인자 값을 꺼낸다. GraphicsWindow.

DrawResizedImage의 첫 번째 인자는 StrV이며, 이 타입인지를 확인하고 값을 꺼내온다. 그리고 리턴해야 하는 값이 있다면 결과를 StrV, DoubleV, ArrayV 중 맞는 타입으로 결과 값을 넣어 리턴한다.

4.2 확장 라이브러리 변수 작성

GraphicsWindow.Width와 같은 라이브러리 변수를 읽고 쓸 때도 마찬가지로 라이브러리 이름의 문자열과 변수 이름의 문자열을 자바 리플렉션 방법을 통해 실제 라이브러리 변수를 참조해 읽기와 쓰기를 진행한다. 그러나 라이브러리 변수를 다룰 때 라이브러리 함수를 호출하는 것과 다른 점이 있다. 스몰베이직 프로그램에서 GraphicsWindow.Width에 폭을 지정하면 윈도우 크기가 바로 변경된다. 이 동작을 지원하기 위해서 라이브러리에 반드시 notifyFieldAssign 함수를 포함시켜 프로그램 해석기에서 이 라이브러리의 변수를 갱신한 다음 이 함수를 호출하여 갱신한 변수 이름을 알려주도록 설계하였다. 이 함수가 호출되면 인자로 받은 갱신된 변수 이름을 확인하고 윈도우 크기를 변경하는 액션을 취할 수 있다. 라이브러리 변수를 읽기 전에는 notifyFieldRead 함수를 두어 프로그램 해석기에서 읽을 변수 이름을 전달하면 라이브러리에서 이 변수의 값을 미리 설정해둔다. 예를 들어, 스몰베이직 프로그램에서 날짜를 확인할 때 해석기에서 Clock.Date 변수를 읽기 전에 Clock 라이브러리의 notifyFieldRead를 호출하여 현재 날짜로 설정해둔 다음 이 변수를 읽는다. 변수를 다루기 위한 라이브러리 필수 함수인 notifyFieldRead와 notifyFieldAssign은 다음 (그림 26)과 같이 작성하면 된다.

```
package com.coducation.smallbasic.lib;

public class GraphicsWindow {
  public static void notifyFieldAssign(String fieldName) {
    if (fieldName.equalsIgnoreCase(“Width”)) {
      double width = ... set Width value set ...;
      ... change the width of a window ...
    }
    ...
  }
  public static void notifyFieldRead(String fieldName) {
    if (fieldName.equalsIgnoreCase(“Width”)) {
      Width = set a value for program to read after this call ;
    }
    ...
  }

  public static Value Width;
  ...
}
```

그림 26 확장 라이브러리 변수 작성 방법

라이브러리를 사용할 때마다 자바 리플렉션을 사용하는 것이 다소 비효율적일 수 있으나 코딩 교육의 목표로 작성한 프로그램의 경우 학생들에게 전혀 방해 요소가 되지 않는다. 마이스몰베이직에서 실행한 프로그램이 느려 실습 진행이 어려운 적은 없었다. 향후 연구로 해석기 구조를 컴파일러 구조로 발전시켜 스몰베이직 프로그램으로부터 자바 프로그램을 생성하면 자바 리플렉션을 사용하지 않고 라이브러리를 사용할 수 있을 것이다.

기존의 스몰베이직 표준 라이브러리를 이와 같은 구조에 맞추어 작성하였으며, 이

는 (표 16)과 같다.

표 16 스몰베이직 표준 라이브러리

라이브러리	설명
Array	Array functions
Clock	System clock functions
Controls	Button, TextField, MultiLine TextField
Desktop	Desktop wallpaper
Dictionary	Online dictionary
Flickr	Access to Flickr photo service
File	File and directory management
GraphicsWindow	Graphics functions and variables
ImageList	Image management
Math	Math functions
Mouse	Mouse cursor and button properties
Network	File download
Program	Program execution controls
Shapes	Movable graphics figures
Sound	Sound play, stop, pause functions
Stack	Stack functions
TextWindow	Text-console based input/output
Text	Text manipulation
Timer	Timer functions
Turtle	Turtle graphics

그리고 앞에서 설명한 방식의 라이브러리 구조에 맞추어 작성한 확장 라이브러리 사례는 (표 17)과 같다. Assert, List, Tree, Graph 라이브러리는 처음부터 새로 개발하였고, Chart, Database, Facebook, Hamster, Video, Weka 라이브러리는 관련 오픈소스 소프트웨어를 활용하여 개발하였다.

표 17 스몰베이직 확장 라이브러리

라이브러리	설명
Assert	Assertion functions
Chart	Chart drawing functions
Database	SQLite3-based Database functions
Facebook	RestFB-based Facebook functions
Graph	Graph functions
Hamster	Hamster Robot control functions
List	List functions
Tree	Tree functions
Video	Video player functions
Weka	Weka-based learning functions

제 4 장 스몰베이직 확장 및 논의사항

제 1 절 스몰베이직 해석기 확장

앞 절에서 기존의 스몰베이직 코딩 환경과 호환되는 스몰해석기를 만들기 위해 필요한 주제들을 논의하였다. 이 절에서 이 호환성을 최대한 유지하면서 언어, 라이브러리, 코딩 환경을 확장 개발한 내용을 설명한다.

1.1 마이스몰베이직 디버거

마이크로소프트 스몰베이직 코딩 환경에서 디버깅 기능을 제공하지 않아 초보자가 스몰베이직 프로그램의 제어 흐름이나 특정 시점의 변수 값을 확인하는데 어려움이 있었다. 디버거를 통해 스몰베이직 프로그램을 더 잘 이해할 수 있는 환경을 제공하기 위해 마이스몰베이직 프로젝트에서 소스프로그램 수준의 디버거를 개발하였다[12].

1.1.1 사용자 관점에서의 디버거

사용자는 프로그램의 오류를 해결하거나 프로그램의 구조를 파악하기 위해 디버거를 사용한다. 그러나 기존의 마이크로소프트의 스몰베이직에서는 디버거를 제공하지 않아 TextWindow를 이용하여 값을 하나하나 출력하여 확인해야하는 유사 디버깅 과정을 거쳐야 했다. 마이스몰베이직은 Java로 구현되어 Java Platform Debugger Architecture(JPDA)를 통해 디버거를 구현할 수 있었다.

사용자는 (그림 27)에 표시된 것과 같이 마이스몰베이직의 디버거를 통해 디버깅 기능을 사용할 수 있다. 디버깅 모드를 시작하기 전에 사용자는 프로그램의 라인 번호 옆의 파란 바에 중단점을 설정해야 하며, 중단점은 여러 개 설정할 수 있다. 다음으로 디버그 버튼을 클릭하면 디버깅 모드가 시작된다. 디버깅 모드에서는 한 문장씩만 실행하거나 다음 중단점까지 실행할 수 있으며, 디버깅 모드를 종료하고 싶을 때는 종료 버튼을 통해 디버깅 모드를 종료할 수 있다.

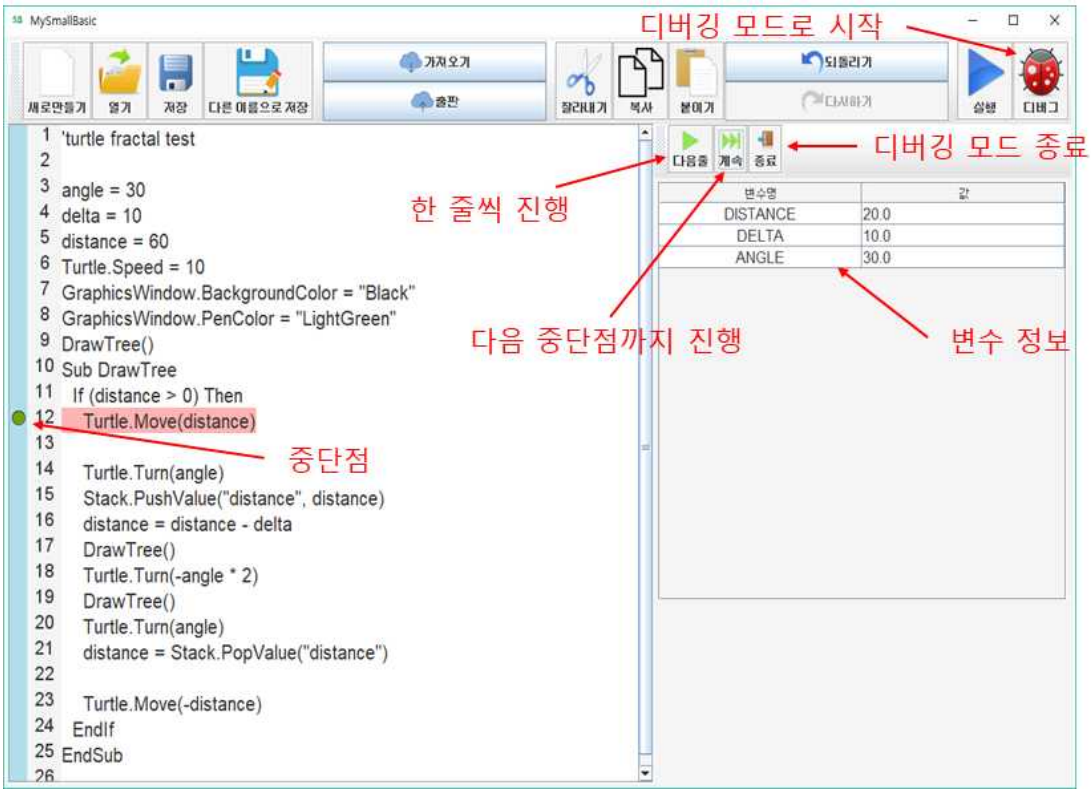


그림 27 마이스몰베이직 디버거 모드

그림 28 스몰베이직 프로그램 디버깅 예시

(그림 27)은 입문자가 이해하기 힘든 재귀함수 호출을 활용한 프로그램을 디버깅

하는 예시이다. (그림 27)에서 보여주는 27라인의 프로그램은 (그림 28)과 같은 트리를 그리는 프로그램이다. 재귀는 매우 중요하지만 처음 프로그램을 배우는 입문자에게는 이해하기 힘든 개념이다. 하지만 디버거 기능을 사용한다면 재귀에 대한 개념을 이해하는데 도움이 된다. 프로그램의 특정 포인트에 브레이크 포인트를 설정한 후, 디버그 버튼을 누르면 (그림 27)과 같은 화면을 볼 수 있다. 다음줄 버튼을 클릭하면 한 줄 진행하며 오른쪽 패널의 변수 값이 변하는 것을 확인할 수 있다. 다음 브레이크 포인트로 돌아가서 값을 확인하고 싶다면 계속 버튼을 클릭함으로써 함수가 한번 실행된 수의 화면을 확인할 수 있다. 디버깅 과정을 끝내고 싶다면 종료 버튼을 눌러 프로그램 종료가 가능하다.

1.1.2 마이스몰베이직 디버거 구조

마이스몰베이직 디버거의 구성은 (그림 29)와 같다. 디버거 자체를 실행하는 자바가상기계(Java Virtual Machine)과 디버깅 대상이 되는 스몰베이직 프로그램을 실행하는 자바가상기계는 별도의 프로세스에서 동작한다. 이 두 자바가상기계는 JDWP(Java Debug Wire Protocol)을 통해서 스몰베이직 프로그램 해석기의 실행 과정을 마이스몰베이직 IDE에서 제어하고 모니터링 할 수 있다. 디버깅 대상 프로그램의 실행 과정을 제어하고 모니터링하기 위해서 자바가상기계 툴 인터페이스(JVM Tool Interface)가 필요하다. 이 인터페이스는 Java JDK에 포함되어 설치되는 tools.jar 파일에서 제공한다.

마이스몰베이직 IDE의 디버거 기능은 JDI(Java Debug Interface)를 통해서 Java 프로그램 수준에서 디버깅 대상 프로그램의 실행을 제어하고 모니터링 할 수 있다. 마이스몰베이직의 파싱 단계에서 분석한 스몰베이직 프로그램의 줄 번호 정보를 Java 기반 해석기와 연결 지어 코딩 환경에서 소스 프로그램의 원하는 위치에 정지점을 설정하고 디버깅 모드로 실행하는 방법을 제공한다. 한 줄씩 실행하거나 정지점까지 실행하는 방법을 제공하고, 실행하면서 변경되는 변수 값을 코딩 환경에서 직접 확인할 수 있는 기능을 제공한다.

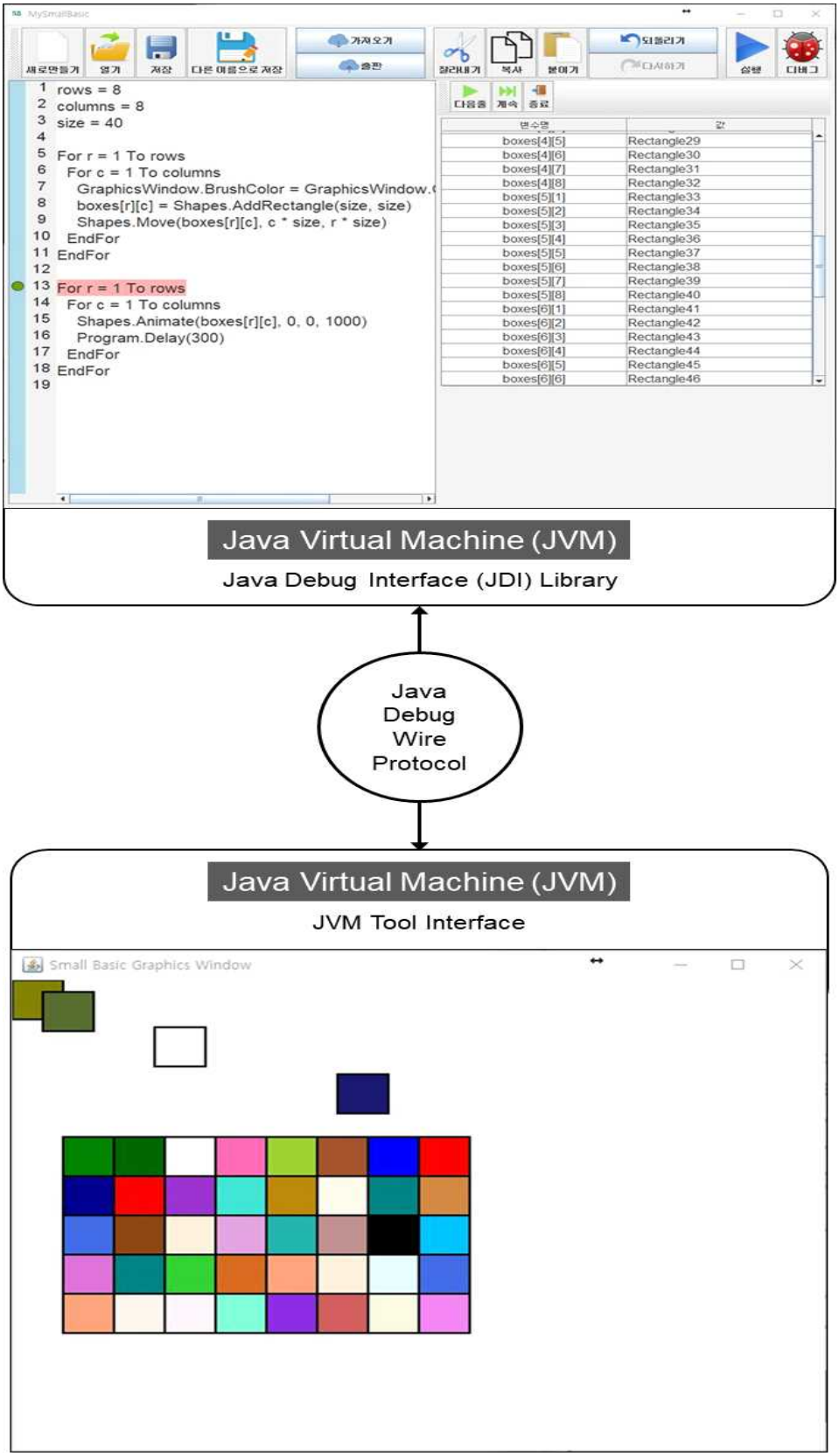


그림 29 마이스몰베이직 디버거 구성도

다음은 마이스몰베이직의 상세 동작 과정이다.

(1) 마이스몰베이직 환경에서 원하는 문장의 줄 번호 앞을 더블 클릭하여 녹색 원 모양의 정지점을 표시한다.
(2) 디버그 버튼을 누르면 MySmallBasicDebugger 클래스 객체를 생성한다.
(3) MySmallBasicDebugger 클래스에서 ProcessBuilder를 통해 디버깅할 대상 프로그램을 실행할 프로세스를 만든다. 디버깅 모드의 자바가상기계 실행 옵션은 “-agentlib:jdwp=transport=dt_socket.address=localhost:7070,server=y.suspend=y”과 같이 지정된다. 위 옵션처럼 소켓통신을 통해 마이스몰베이직 디버거 프로세스와 마이스몰베이직 대상 프로그램을 실행할 프로세스를 연결한다.
(4) JDI 인터페이스를 다루기 위해 오픈소스 JDIScript를 활용하여 jdiScript 객체를 생성한고 (3)에서 만든 디버깅 대상 프로세스를 붙인다.
(5) jdiScript를 통해 디버깅할 프로세스가 com.coducation.smallbasic.Eval 클래스의 eval(BasicBlockEnv, Env, Stmt) 메소드를 호출하여 스몰베이직 프로그램의 각 문장을 실행할 때마다 멈추도록 이벤트를 정의한다.
(가) 마이스몰베이직 프로그램 문장의 해당 줄 번호에 정지점이 설정되어 있는지 검사한다.
(나) 현재 실행중인 스몰베이직 프로그램의 변수 정보를 가져온다.
(다) MySmallBasicDebugger에서 정의한 정지점에 대한 이벤트가 발생하면 멈춘 상태에 대한 변수정보를 넘겨 GUI를 업데이트한다.
(6) 마이스몰베이직 프로그램의 현재 실행 문장에 중지점이 설정되어 있지 않더라도 일단 Eval의 메소드 eval을 호출한 직후에 대상 프로그램이 정지하므로 이 때 생성된 콜백 함수를 리턴함으로써 실행을 계속하도록 만든다.

마이스몰베이직 디버거를 구현하기 위해 필요한 클래스를 개발했다. 마이스몰베이직 프로그램의 변수 값들을 테이블로 보여주기 위한 MonitoringTable 클래스, 디버거에 제공할 기능을 가지고 있는 추상 클래스 MySmallBasicDebuggerModel, 이 추상클래스를 구현하여 디버깅 중인 프로그램을 관리하고 디버거 클라이언트와 정보를 주고 받는 MySmallBasicDebugger 클래스가 있다. 또 디버거를 사용하는 클라이언

트에게 제공할 기능을 가지고 있는 MySmallBasicDebuggerClientModel 인터페이스, 마이스몰베이직 그래픽 사용자 인터페이스와 MySmallBasicDebuggerClientModel을 구현한 MySmallBasicGUI 클래스가 있다.

오픈소스 프로젝트로 개발하였기 때문에 디버거 개발과 같은 코딩 환경의 확장이 용이하였다. 왜냐하면 디버거 개발을 위해 마이스몰베이직 프로젝트의 해석기 내부 구조를 반드시 알아야하기 때문이다. 특히 구문 분석(Parsing)에서 얻은 추상구문트리(abstract syntax tree)와 변수 환경에 관한 정보는 디버거를 구현할 때 반드시 필요하다.

1.2 확장 라이브러리

3장의 4절에서 설명한 라이브러리 구조를 토대로 초보자의 흥미를 높일 수 있는 라이브러리를 개발하였다. 그 중 햄스터 로봇 제어 라이브러리, Weka 학습 라이브러리를 설명하며, 라이브러리 구조 문서를 따라 만들어진 기타 확장 라이브러리를 소개한다.

1.2.1 햄스터 로봇 제어 라이브러리

햄스터 로봇 제어 라이브러리[8]인 Hamster는 코딩 교육에 피지컬 컴퓨팅[9,10] 개념을 도입하여 흥미를 높이기 위해서 추가되었다. 햄스터 로봇의 모터를 스몰베이직 프로그램을 통해 제어하고 부착된 센서 정보를 받아 상황에 따라 상호작용한다. 햄스터 라이브러리는 모터 속도를 제어하는 Wheel 함수와 방향을 전환하는 Rotate함수, 장애물을 감지하는 IsObstacleInDistance 함수, LED 제어, 소리 제어, 센서 제어, 확장키트 제어 함수 등 총 71개의 함수로 구성되어 있다.

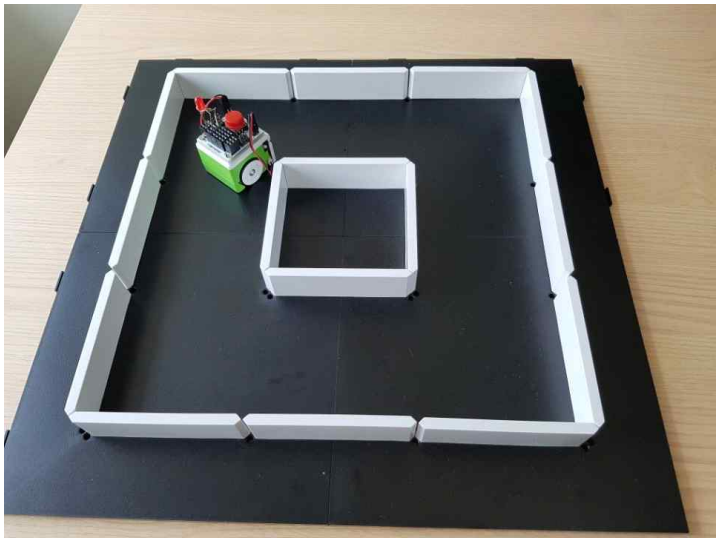


그림 30 스몰베이직 프로그램으로 제어하는 햄스터 로봇

(그림 30)은 확장 라이브러리를 사용한 스몰베이직 프로그램으로 햄스터 로봇을 제어하여 장애물을 만나면 오른쪽으로 회전하는 데모를 보여준다. 이 프로그램을 사용하여 햄스터로 하여금 미로를 탈출하도록 하는 초보자의 흥미를 끄는 코딩 예제를 (그림 31)과 같이 만들었다.

```
While “True”
 Hamster.Wheel(40)
 If Hamster.IsObstacleInDistance(35) = “True” Then
  Hamster.Rotate(30)
  While “True”
   If Hamster.IsObstacleInDistance(35)=“False” Then
    Goto ESCAPE
   EndIf
  EndWhile
 EndIf
ESCAPE:
EndWhile
```

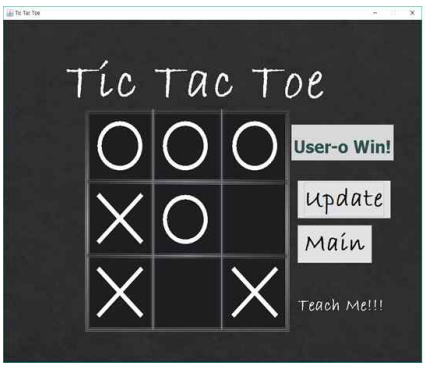
그림 31 스몰베이직 프로그램 예제: 햄스터 로봇 제어

1.2.2 Weka 학습 라이브러리

Weka 학습 라이브러리[11]는 데이터마이닝 Weka 오픈소스 소프트웨어를 기반으로 만들어진 확장 라이브러리이다. Weka에서는 Java 기반 라이브러리를 제공하기 때문에 이를 이용하여 인공지능 프로그램을 작성해볼 수 있지만, 어려운 개념을 필요로 하기 때문에 마이스몰베이직 라이브러리로 확장했다.

마이스몰베이직의 Weka 라이브러리는 데이터를 분석하고 목표 변수를 예측하는 Classify 함수와 학습한 결과를 Arff 파일에 업데이트하는 Update 함수 2개를 가진다. Classify는 기반 데이터와 학습할 데이터의 배열을 인자로 받는다. 데이터 집합을 분석하여 목표 변수의 값을 예측하여 이를 반환한다. Update는 업데이트할 데이터의 배열과 기존의 학습 데이터 경로를 인자로 받는다. 기존의 학습 데이터가 담긴 Arff 파일에 업데이트할 데이터를 추가한다.

이 Weka 학습 라이브러리를 이용하여 틱택토 프로그램을 작성하였다. 시작화면은 시작버튼, 학습버튼, 도움말버튼, 나가기 버튼으로 구성되어 있다. 시작화면에서 학습 버튼을 클릭하면 (그림 32)의 왼쪽화면으로 전환된다. Teach에서는 승부가 나기 전까지의 판 상황을 arff 파일에 업데이트 하는 것이다. 사용자 혼자서 임의로 게임을 진행하다가 승부가 나게되면 Update 버튼을 눌러 Weka의 Update를 호출하는 코드는 (그림 32)의 오른쪽과 같다.



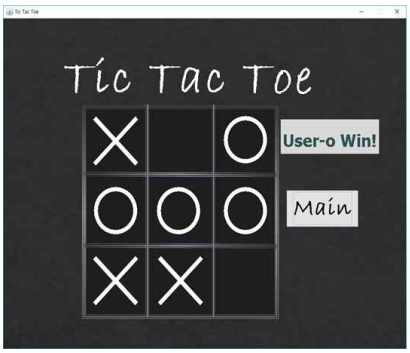

```
' 승부가 나기 전까지 매번 판이 클릭될 때마다
' 판의 상황을 저장한 2차원 배열의 history를
' 학습파일인 arff 파일에 업데이트

Sub Learning
  If result = 1 Or result = 2 Then
    For i = 0 To count  ' count는 둔 횟수
      Weka.Update(arff, hBoard[i])
    EndFor
  EndIf
EndSub
```

그림 32 Weka 라이브러리를 이용한 틱택토 예제: 학습

충분한 학습과정이 끝났다면 메인화면으로 돌아가 시작 버튼을 클릭해 학습된 데이터와 겨룰 수 있다. 사용자의 턴이 지나고 컴퓨터의 턴이 돌아오면, 컴퓨터는 빈 곳에 미리 두어 이길 확률을 확인한다. 그 중 이길 확률이 가장 높은 경우의 빈 자리에

자신의 패를 업데이트 한다. (그림 33)은 사용자와 컴퓨터가 겨룰 때의 화면(왼쪽)과 Classify를 호출하는 코드(오른쪽)을 보여준다.




```
' 다음에 둘 차례에 판이 빈 곳이 있으면
' 미리 두어 본 다음 Weka.Classify를 호출하여
' 그 자리에 두었을 때 이길 확률을 확인

If board[i] = "b" Then
  cborad[i] = "x"     'cboard는 board를 copy한 것
  positive = Weka.Classify(arff, cboard)

  ' positive가 높은 자리를 가지고 있도록 하는 코드
```

그림 33 Weka 라이브러리를 이용한 틱택토 예제: 대결

1.2.3 기타 확장 라이브러리

위에서 자세히 다루지 않았지만 Chart, Graph, List, Tree는 스몰베이직을 통해 자료구조에 대한 이해도를 높이기 위해 개발한 라이브러리이다. 이 뿐만 아니라 Video 라이브러리를 개발하여 제공함으로써 초보자는 다양한 컨텐츠를 다룰 수 있다. Database와 Facebook 라이브러리는 베트남 HUST 대학 2팀(총 6명)이 GitHub 프로젝트 사이트에 제공하는 라이브러리 구조 문서를 보고 개발하여 기여한 확장 라이브러리이다.

이와 같이 확장 라이브러리들을 개발한 경험을 바탕으로 새로운 라이브러리를 개발하고 추가하더라도 이미 개발된 스몰베이직 해석기와 코딩 환경에 영향을 주지 않는 독립적인 라이브러리 구조를 갖추었음을 확인하였다. 이러한 구조를 활용하면 누구나 마이스몰베이직 오픈소스 소프트웨어 프로젝트에 새로운 라이브러리를 쉽게 기여할 수 있을 것이다.

1.3 스몰베이직 언어의 확장: Assert 문

오픈소스 소프트웨어 프로젝트의 속성상 다수의 개발자가 코드를 기여할 때마다 CI 기반으로 빌드하고 테스트할 필요가 있다. 마이스몰베이직의 코딩 환경은 스몰베이직 프로그램들을 활용하여 회귀테스트를 한다. 새로운 코드가 프로젝트에 추가될 때마다 이 과정을 자동화하기 위해 프로그램에 assert 문을 작성하는 방법을 구현하였다.

```
a = File.WriteContents("tmp.txt", "abcd")
b = File.ReadContents("tmp.txt")
c = File.WriteContents("tmp.txt", "\r\nabcd")
d = File.ReadContents("tmp.txt")

'@assert a = "SUCCESS"
'@assert b = "abcd"
'@assert c = "SUCCESS"
'@assert d = "\r\nabcd"
```

스몰베이직에서 작은따옴표로 시작해서 줄 바꿈 문자까지 주석에 해당한다. 테스트 담당자가 주석에 확장된 키워드 @assert와 조건식을 작성하면 마이스몰베이직 해석기가 이 주석이 있는 줄을 자동으로 Assert 라이브러리의 함수를 호출하는 “assert (조건식)”으로 바꾼다. 이 줄이 실행될 때 이 조건식을 계산하여 참이 되면 다음 줄로 이동하고 거짓이면 그 위치에서 에러 메시지를 출력하고 실행을 종료한다. 위 프로그램에서 파일을 읽고 쓰기를 두 차례씩 반복하면 변수 a, b, c, d에 지정한 문자열이 저장되어 있어야 한다. 이를 assert문으로 작성하였다.

주석으로 assert문을 작성하기 때문에 기존 마이크로소프트 코딩 환경에서 assert 문을 포함한 프로그램을 수정하지 않고도 그대로 실행할 수 있다.

제 2 절 코딩 교육 활용 사례

이 절에서 마이스몰베이직을 2018학년도 봄 학기 컴퓨터과학적사고 과목에서 실습 용도로 활용하였을 때 얻은 경험을 설명한다. 이 프로젝트는 아직 초기 단계이므로 코딩 교육의 효과를 정량적으로 판단하기에 부족하다. 따라서 이 절에서는 활용 경험을 정성적으로 서술한다.

2.1 컴퓨터과학적사고 강의 개요

표 18 컴퓨터과학적사고 과목의 주요 주제

	Topics
1	Computational Thinking - What is computational thinking - Understanding effective procedures
2	Information Representation - Bits - Number, String
3	Logic - Logic application in daily lifes - Symbolic logic and inference rules
4	Computer Programming - Elements of programming languages - How to design effective procedures
5	Information Organization - List, Tree, Graph - Algorithms

컴퓨터과학적사고 과목의 주요 주제는 컴퓨터과학적 사고, 정보표현, 논리, 기초 프로그래밍, 기초 자료구조 및 알고리즘이다. (표 18)에 이 과목에서 한 학기 동안 다룬 5가지 주제를 요약하였다.

(표 18)에서 나열한 주제의 강의를 진행하면서 전반부는 주로 종이와 연필을 활용하여 실습을 진행하였고 후반부에 특히 4번째 주제 이후부터 본격적으로 스몰베이직을 활용한 코딩 실습을 진행하였다.

각 주제에 대해 아래 (표 19)의 스몰베이직 프로그램을 학생들로 하여금 마이스몰베이직을 사용하여 전체를 작성하거나 또는 일부를 채워 완성하는 방식으로 실습을 진행하였다.

표 19 컴퓨터과학적사고 과목에서 설계한 스몰베이직 프로그램 예제

	SmallBasic Programs	Related Topics
1	Newton Square Root Calculation	Computational Thinking
2	Rock-Scissors-Paper Game	Information Representation
3	Course Registration Data Analysis	Basic Programming
4	Tic-tac-toe	How to design an effective procedure
5	Rush Hour Game	Information Representation
6	Josephus Problem	List
7	Word Auto Completion	Tree
8	Uncovering a Hidden Map	Graph
9	Finding Subway Directions	Dijkstra's Shortest Path Algorithm

2.2 예제 프로그램을 통한 강의 진행

이 절에서는 컴퓨터과학적사고 강의에서 설계한 스몰베이직 프로그램 예제를 활용한 사례와 강의의 진행방식에 대해서 설명한다. 정보표현을 다룰 수 있는 Rush Hour Game과 자료구조 그래프를 활용하여 가장 짧은 경로를 찾는 다익스트라 알고리즘을 이용한 Finding Subway Directions 예제 프로그램을 소개한다.

2.2.1 정보표현 예제

(표 19)에서 설계한 예제 프로그램을 통해 강의 중 5번째 프로그램 Rush Hour Game은 복잡한 주차장에서 차를 움직여 빨간색 차를 바깥으로 빠져나올 수 있도록 하는 게임이다. 이는 학생들이 차에 대한 데이터를 어떻게 표현할 것인지에 대해 배울 수 있는 예제 프로그램이다. Rush Hour Game의 차는 대각선으로는 움직일 수 없으며, 좌우 또는 상하로만 움직일 수 있다는 규칙이 있다. 학생들에게 Rush Hour Game의 초기상태인 (그림 34)를 보여주며, 차에 대한 데이터를 표현하는 3가지 방법에 대해 설명했다.

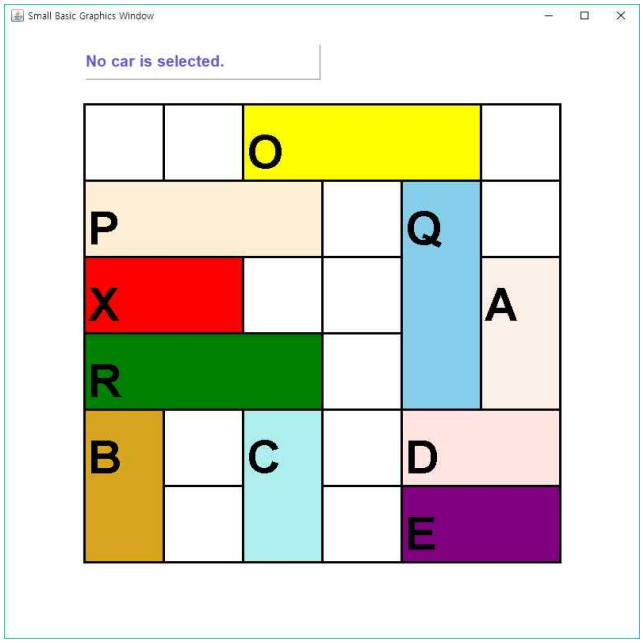


그림 34 Rush Hour Game 초기상태

- 첫 번째 정보표현 방법은 각 블록의 시작 좌표와 마지막 좌표를 입력하는 것이다. 예를 들어, (그림 34)에 있는 블록 O와 Q를 표현해보자. 좌측 상단은 1부터 시작하며 가로축은 x좌표를, 세로축은 y좌표를 의미한다. 블록 O는 (3, 1)에서 시작해 (6, 1)에서 끝난다. 블록 Q는 (5, 2)에서 시작해 (5, 5)에서 끝난다. 이를 스몰베이직에서 지원하는 배열로 작성을 해본다면, 다음과 같이 표현할 수 있다.

```
01: ‘ Block O                          07: ‘ Block Q
02: rushhour[0][“name”] = “O”          08: rushhour[1][“name”] = “Q”
03: rushhour[0][“startX”] = 3          09: rushhour[1][“startX”] = 5
04: rushhour[0][“startY”] = 1          10: rushhour[1][“startY”] = 2
05: rushhour[0][“endX”] = 6            11: rushhour[1][“endX”] = 5
06: rushhour[0][“endY”] = 1            12: rushhour[1][“endY”] = 5
```

- 두 번째 정보표현 방법은 2차원 배열에 맵 전체 정보를 입력하는 것이다. (그림 34)를 통해 맵은 6*6의 크기를 가지는 것을 확인할 수 있다. 이 맵 전체 중에서 위에서 3번째 줄까지만 스몰베이직에서 지원하는 배열로 작성을 해본다면 다음과 같이 표현할 수 있다. “,”은 아무것도 없는 빈 블록임을 의미하며, 각 블록을 차지

한 블록 이름을 작성하는 방식이다.

```
01: ‘ Rush Hour Map
02: rushhour[0][0] = “.”
03: rushhour[0][1] = “.”
04: rushhour[0][2] = “O”
05: rushhour[0][3] = “O”
06: rushhour[0][4] = “O”
07: rushhour[0][4] = “.”
08:
09: rushhour[1][0] = “P”
10: rushhour[1][1] = “P”
11: rushhour[1][2] = “P”
12: rushhour[1][3] = “.”
13: rushhour[1][4] = “Q”
14: rushhour[1][5] = “.”
15:
16: rushhour[2][0] = “X”
17: rushhour[2][1] = “X”
18: rushhour[2][2] = “.”
19: rushhour[2][3] = “.”
20: rushhour[2][4] = “Q”
21: rushhour[2][5] = “A”
```

- 세 번째 정보표현 방법은 각 블록의 시작좌표와 그 블록의 길이, 수직/수평의 정보를 입력하는 것이다. 첫 번째 정보표현 방법과 동일하게 블록 O와 Q를 표현해 보자. 블록 O는 (3, 1)에서 시작해 길이가 3이며 수평으로 그려져야 한다. 블록 Q는 (5, 2)에서 시작해 길이가 3이며 수직으로 그려져야 한다. 이를 스몰베이직의 배열로 작성해보면 다음과 같이 표현할 수 있다.

```
01: ‘ Block O
02: rushhour[0][“name”] = “O”
03: rushhour[0][“x”] = 3
04: rushhour[0][“y”] = 1
05: rushhour[0][“length”] = 3
06: rushhour[0][“direct”] = “H”
07: ‘ Block Q
08: rushhour[1][“name”] = “Q”
09: rushhour[1][“x”] = 5
10: rushhour[1][“y”] = 2
11: rushhour[1][“length”] = 3
12: rushhour[1][“direct”] = “V”
```

이와 같이 세 가지 방법 중 하나를 이용해서 정보를 표현하여 실행한다면 (그림 34)의 화면을 통해 Rush Hour Game을 즐길 수 있다. 이 세 가지 방법 중 마지막 방법을 통해 데이터를 표현한 코드는 부록B에서 확인할 수 있다.

2.2.2 다익스트라 알고리즘

(표 19)의 9번째 프로그램 Finding Subway Directions는 다익스트라 알고리즘(Dijkstra Algorithm)을 통해 출발점에서 도착점으로 가는 최소 경로를 구하는 것이다. 학생들에게 다익스트라 알고리즘에 대해서 상세하게 설명해주고, 다익스트라 알고리즘의 일부분을 비워두고 이를 채우는 방식으로 진행하였다.

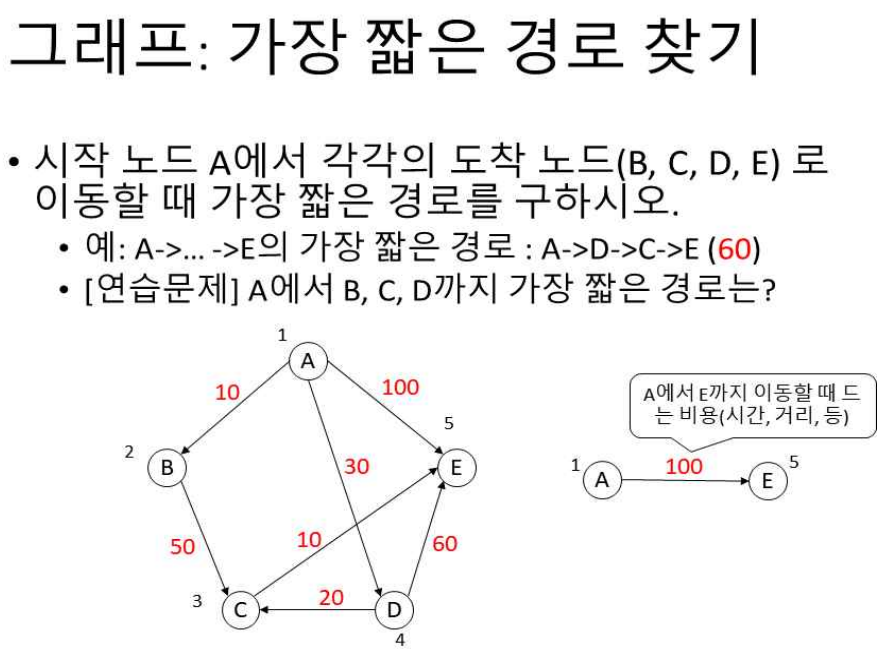


그림 35 다익스트라 알고리즘 연습문제

먼저, 다익스트라 알고리즘에 대해서 설명하기 전에 학생들에게 (그림 35)와 같은 연습문제를 보여주며 이 문제를 풀어볼 수 있는 시간을 주었다. 충분한 시간을 줌으로써 학생들이 답을 찾아내었고, 답을 찾아가는 과정을 설명해주었다. 그리고 이 과정으로 가장 짧은 경로를 찾는 것이 다익스트라 알고리즘임을 알려주며 각 과정을 다시 짚어보았다.(그림 36)

그래프: 가장 짧은 경로 찾기

- 다익스트라 알고리즘
  - 입력: 그래프, 시작 노드
  - 출력: 시작 노드~각 노드로 가는 최소 경로 및 비용
  - 절차:
    1. 시작 노드의 비용을 0, 나머지 노드의 비용을 ∞로 초기화
    2. 빗금을 칠하지 않는 노드가 있는 동안 반복
       2-1. 빗금을 칠하지 않은 노드들 중 최소 비용의 노드(u)를 선택
       2-2. 선택한 노드 u에서 분기하는 각 노드(w)의 비용에 대해:
            2-2-1. 만일 (A~u 비용 + u->w의 비용) < (A~w 비용)이면
                 2-2-1-1. A~w 비용을 더 적은 비용으로 수정
                 2-2-1-2. u->w를 수정된 비용 옆에 기록
       2-3. 노드 u를 빗금으로 칠함

그림 36 다익스트라 알고리즘 절차

다익스트라 알고리즘의 초기화 부분, 최소 비용의 노드를 찾는 부분 그리고 선택된 노드에서 분기하는 각 노드 중 최소의 비용을 가지게 되는 노드를 찾는 부분을 주석 처리하여 채워보도록 코드를 (그림 37)과 같이 제공하였다. 스몰베이직으로 구현한 다익스트라 알고리즘은 부록B에서 확인할 수 있다.

```
Sub dijkstra
  ' 1. 시작 노드 비용을 0, 나머지 노드 비용을 정수 최대수로 초기화
  ' 2. 아직 가보지 않은 노드 중 최소 비용의 노드(u) 선택
  ' 3. 선택된 노드(u)에서 분기하는 각 노드(w)에 대해 최소의 비용 찾기
  For j = 1 To map[u]["count"]
    w = map[u][j]
    ' 선택된 노드에서 분기하는 각 노드에 대해
    ' 최소의 비용을 갖는 노드를 찾는 조건문을 작성하세요.
      map[w]["cost"] = map[u]["cost"] + map[u]["edge"][j]
      map[w]["prev"] = u
    EndIf
  ...
EndSub
```

그림 37 컴퓨팅과학적사고 프로그램 예제: Finding Subway Directions

2.3 교육용 언어로써의 마이스몰베이직

마이스몰베이직을 컴퓨터과학적사고 강의 및 실습에 적용한 결과 다음과 같은 흥미로운 사항을 발견할 수 있었다. 첫째, 스몰베이직이 배우기 쉬운 프로그래밍언어라는 장점을 다시 한 번 확인할 수 있었다. 강의 시간에 스몰베이직을 직접 가르치는 것을 지양하였고 튜토리얼 문서를 제공하고 스스로 공부하도록 하였다. 다만 강의 시간에 스스로 공부하였는지 확인하는 퀴즈를 진행하였다.

For나 While과 같은 반복문을 작성하는 것은 어려워하지만 If와 Goto문에 매우 쉽게 적응하는 것을 알 수 있었다. 정식 프로그래밍 과목이라면 전자의 반복문을 사용하는 것이 바람직하지만 그렇지 않다면 If와 Goto만을 사용하여 제어 흐름을 구성하는 것도 가능한 방법이라 판단된다.

전역변수만을 사용하되 서브루틴의 인자를 두지 못하도록 설계한 스몰베이직의 특징 때문에 학생들은 서브루틴을 자동으로 돌아오는 분기문 Goto로 매우 쉽게 이해할 수 있었다. 다른 프로그래밍언어의 경우 인자 전달에 대한 설명을 했어야 하겠지만 마치 클래스와 객체 개념과 같이 초기 코딩 교육에는 필요한 요소는 아니다.

둘째, 학생들의 흥미를 끌기 위해 스몰베이직 표준 라이브러리로 충분하였다. 실습에서 주로 그래픽스 라이브러리를 많이 사용하였고 그 다음으로 텍스트 라이브러리를 사용하였다. 리스트, 트리, 그래프에 대한 확장 라이브러리를 새로 만들었지만 배열로 각 자료 구조를 표현하는 방법을 학생들이 직관적으로 잘 이해하였기 때문에 별도의 확장 라이브러리들을 활용하지 않았다.

셋째, 최소의 특징만을 고려한 스몰베이직의 경우조차도 프로그래밍 요소(If, For 등)와 라이브러리 함수와 변수가 너무나 다양하기 때문에 초보자가 어느 것을 사용해서 프로그램을 작성해야 하는지 막막하게 생각하는 경우가 빈번하게 발생한다. 이 문제에 대한 대응 방안이 필요하다. 학생들이 스몰베이직 프로그램을 작성할 때 선택할 수 있는 범위를 한정지어 주는 방법이 있다면 더 쉽게 작성할 수 있을 것이다. 예를 들어 지하철 길찾기 프로그램의 경우 최단거리 찾는 알고리즘의 핵심만 구현하면 나머지 그래픽을 다루는 요소는 미리 제공하는 방식을 사용하였다.

넷째, 마이스몰베이직 프로젝트는 초기 개발 단계이므로 구문 분석 단계나 실행 단계에서 오류가 발생하는 경우 코딩 환경의 사용자에게 보다 쉽게 이 오류를 해결할 수 있도록 자세히 설명하는 방법을 아직 제공하지 못하고 있다. 초보자가 사용하는 코딩 환경을 목표로 하는 만큼 오류 처리에 대한 많은 고려가 필요하다.

제 5 장 결론 및 향후 연구

본 논문에서는 마이크로소프트 스몰베이직을 확장한 오픈소스 소프트웨어 마이스몰베이직을 소개하였다. 본 논문에서 공헌한 점은 다음과 같다.

첫째, PC기반의 다양한 운영체제(맥, 리눅스, 윈도우)에서 실행 가능한 스몰베이직을 개발하였다. 기존의 스몰베이직은 마이크로소프트의 닷넷 프레임워크에 종속되어 윈도우 운영체제에서만 사용이 가능하다는 단점이 있다. 자바를 이용하여 개발함으로써 특정 운영체제에 종속되지 않는 스몰베이직 코딩 환경을 개발하였다.

둘째, 기존의 스몰베이직에서 실행할 수 있었던 프로그램이 동일하게 동작하도록 호환성을 지원한다. 스몰베이직 튜토리얼 문서에서 추출한 26개 프로그램과 복잡한 4개의 프로그램을 실행 및 비교 테스트하여 동일하게 동작함을 확인하였다.

셋째, 스몰베이직 프로그래밍 언어의 파서와 동적 형 변환 의미에 대한 명세를 문서화하였다. 해석기를 구현하기 위해 동적 형 변환을 파악하는 것은 매우 중요하지만 이에 대해 설명하는 문서가 없었다. 마이크로소프트 스몰베이직의 튜토리얼 문서, 예제 프로그램을 통해 역공학 분석하였으며, 이를 통해 해석기를 구현하였다.

넷째, 스몰베이직 그래픽스 아키텍처의 구조를 분석하고 개발하였다. 다양한 그래픽스 라이브러리의 구조를 파악함으로써 기존의 스몰베이직 그래픽스 라이브러리와의 호환성을 보장한다.

다섯째, 스몰베이직에 디버깅 모드를 추가 개발하였다. 디버깅 기능을 추가함으로써, 입문자가 프로그램의 실행과정을 한 문장씩 따라가며 제어흐름을 살피고 변수의 값을 쉽게 확인하여 프로그램을 좀 더 쉽게 이해할 수 있도록 확장하였다.

여섯째, 오픈소스 소프트웨어로 언어 및 라이브러리의 자유로운 확장이 가능하다. 기존 스몰베이직은 비공개 소프트웨어로 교육용 프로그래밍 언어로는 적합하지만, 입문자의 흥미를 이끌만한 라이브러리가 한정적이라는 단점이 있다. 마이스몰베이직은 오픈소스 소프트웨어로 개발되어 입문자의 흥미를 이끌 수 있는 Hamster 로봇 라이브러리, Weka 학습 라이브러리, SQLite 데이터베이스 라이브러리 등을 총 10개의 라이브러리를 추가하였다.

일곱째, 마이스몰베이직을 강의에 적용하여 교육용 프로그래밍 언어로써 적합한지를 확인하였다. 학생들이 마이스몰베이직으로 프로그래밍 해봄으로써 교육용 언어로

써의 스몰베이직이 적합한지를 분석하고 이에 따른 효과를 확인하였다.

이러한 결과를 통해 스몰베이직의 확장이 잘 이루어져 운영체제에 종속되지 않는 교육용 코딩 환경을 가졌음을 알 수 있다.

본 논문의 향후 연구는 다음과 같다.

첫째, 효과적인 교육용 코딩환경을 제공하기 위해서 사용자와 상호작용하는 더욱 다양한 기능이 필요하다. 마치 스크래치에서 사용가능한 블록들이 한정되어 있듯이 작성할 수 있는 범위를 한정지어 제시하는 코딩 환경이 사용자에게도 도움이 될 것이다. 또한 구문 오류나 논리 오류가 발생한 경우 스스로 그 이유를 파악하고 대응할 수 있는 환경을 제공할 필요가 있다.

둘째, 마이스몰베이직의 환경의 기능 개발이 필요하다. 스몰베이직 환경에서 제공하고 있는 자동 코드 완성 기능, 최근에 추가된 웹 브라우저에서의 마이스몰베이직 코딩 환경과 같이 좀 더 확장된 환경을 통해 사용자들이 더 쉽고 편하게 코딩할 수 있을 것이다.

셋째, 마이스몰베이직을 실제 교육용 코딩환경으로 적용한 사례를 정량적으로 측정하고, 코딩 교육 효과를 평가하는 연구가 필요하다. 다른 교육용 프로그래밍 언어 및 환경을 이용했을 때의 교육효과와 비교하는 연구 또한 필요하다.

참고문헌

[1] Microsoft Small Basic [Online], https://smallbasic-publicwebsite.azurewebsites.net.

[2] P. Conrod, Lou Tylee, The Developer's Referemce Guide To Microsoft Small Basic, Kidware Software LLC, 2010.

[3] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten, "The WEKA Data Mining Software: An Update, SIGKDD explorations", Vol. 11, No. 1, 2009.

[4] MySmallBasic [Online], http://github.com/kwanghoon/MySmallBasic.

[5] 최광훈, 김가영, 조문영, 김지용, 박세영, 정승완, 조성모, "오픈소스 소프트웨어 결과보고서", 2018년.

[6] Small Basic Tutorial Documents [Online], https://smallbasic-publicwebsite.azurewebsites.net/Pages/Tutorials/Tutorials.aspx.

[7] S. E. Ganz, D. P. Friedman, M. Wand, "Trampolined Style", In Proceedings of the Fourth ACM SIGPLAN International Conference on Functional Programming(ICFP 1999), pp. 18-27, Sept. 1999.

[8] 박세영, 조문영, 최광훈, "교육용 로봇 프로그래밍 위한 스몰베이직 라이브러리 설계 및 구현에 관련 연구", 한국정보처리학회 춘계학술발표대회 논문집, Vol. 24, No. 1, pp. 399-402, 2017년 4월.

[9] 서동수, "피지컬컴퓨팅의 개념과 기술적 기초", 한국디자인학회 2006 가을 학술발표대회 논문집, p270-271, 2006.

[10] 박정호, "초등학교에서 로봇을 활용한 STEAM 교육의 적용 연구", 한국컴퓨터정보학회 논문지, Vol. 17, No. 4, pp. 19-22, 2012년 4월.

[11] 김지용, 정승완, 조성모, 최광훈, "스몰베이직 언어 기반 교육용 인공지능 프로그램 작성을 지원하는 라이브러리 설계 및 구현에 관한 연구", 한국정보처리학회 춘계학술발표대회 논문집, Vol. 24, No. 1, pp. 694-696, 2017년 4월.

[12] 조문영, 최광훈, "스몰베이직 프로그램 디버거 설계 및 구현에 대한 연구", KIISE 한국소프트웨어 종합학술대회 논문집, pp. 2195-2197, 2017년 12월.

A Design and Implementation of Programming Language Environment for Coding Education

Gayoung, Kim

Department of Electronics and Cmputer Engineering

Graduate School Chonnam National University

(Supervised by Professor Kwang Hoon Choi)

(Abstract)

This paper proposes implementation of an opensource Small Basic coding environment. In an era where the role of software has become important, elementary, middle and high schools are required to provide more than a certain amount of coding education, and the global trend is to provide coding education to other major students at universities. Amid the attention of the language to be used in coding education, Small Basic is a simple text-based programming language for beginners, that is easy to learn and user-friendly. But the existing Microsoft Small Basic environment is a closed software and has the disadvantage that it is difficult to extends or add the environment. Therefore, it is necessary to study the educational programming language which can expand the language and library while providing a simple and easy environment for beginners like Small Basic.

This paper proposed a new Small Basic coding environment MySmallBasic, which extends to Java for use in Windows, Linux, and Mac. It is compatible with the programs used in the existing Small Basic language and is an open source project, so anyone can participate in the development. As a by-product of this study, this paper provides formal specifications on the

parser and on dynamic typing semantics for Small Basic programming language, that has never been documented.

The proposed MySmallBasic in this paper can be used in various PC based operating systems. The addition of libraries to interest the beginner is free and further enhances understanding of the program's structure by adding debugging capability.

부록A

부록에서는 제안한 마이스몰베이직을 사용하는 방법에 대해서 소개한다. 부록은 제안한 마이스몰베이직을 다운로드 받고 사용하는 방법, 디버거를 사용하는 방법을 설명한다.

제 1 절 마이스몰베이직 다운로드 및 사용

마이스몰베이직을 다운받아 사용하는 것은 간단하지만, 마이스몰베이직의 이용 목적에 따라 다운로드 및 실행 방법이 달라 이를 설명하고자 한다. 이용목적에 따라 다운로드 및 실행방법은 다르지만 (그림 38)과 같이 마이스몰베이직 시작 화면은 동일하다.

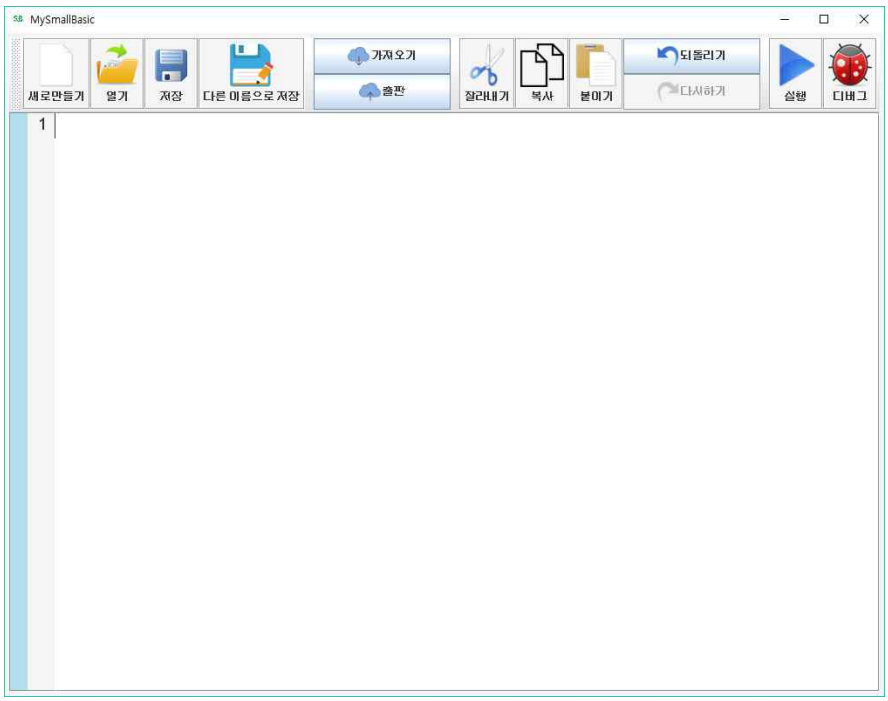


그림 38 마이스몰베이직 시작 화면

1.1 마이스몰베이직을 확장 없이 사용하려는 경우

마이스몰베이직을 확장 없이 사용하려는 경우는 추후의 업데이트를 고려하지 않는다는 가정 하에 다운로드 방법을 설명한다.

먼저, https://github.com/kwanghoon/MySmallBasic 에 접속한다. 오른쪽 상단의 “Clone or download”를 클릭하고 하단의 “Download ZIP”을 클릭한다. 다운로드 받은 폴더 경로로 가서 MySmallBasic-master.zip 파일을 압축 해제한다. 압축이 해제된 MySmallBasic-master 폴더 > MySmallBasic 폴더로 들어가면 run.bat 파일이 있다. 이 파일을 더블클릭하면 간단하게 마이스몰베이직을 사용할 수 있다.

1.2 마이스몰베이직을 확장하여 사용하려는 경우

마이스몰베이직을 확장하여 사용하려는 경우는 추후 업데이트를 계속적으로 받지만 마이스몰베이직에 기여하지 않는다는 가정 하에 다운로드 방법을 설명한다. 이클립스 개발환경을 사용하는 경우, File -> Import -> Git(Project from Git) -> Clone URI를 선택한다. URI 창에 https://github.com/ kwanghoon/MySmallBasic을 입력하고 Next를 누른다. Branch Selection 화면에서도 다시 Next를 누른다. 저장할 디렉토리를 설정해준 다음 Next를 누르고, “Import existing Eclipse projects”를 선택하고 넘어간다. 다운로드는 완료되었고, src/com/coducation/smallbasic/gui/MySmallBasicGUI.java를 실행하면 된다.

1.3 마이스몰베이직을 확장하여 개발에 기여하려는 경우

마이스몰베이직을 확장하여 개발에 기여하는 경우의 다운로드 방법을 설명한다. 먼저 github에 로그인 한 뒤, https://github.com/kwanghoon/MySmall Basic 에 접속하여 오른쪽 상단의 Fork 버튼을 누른다. 이렇게 되면 MySmallBasic이 자신의 repository에 복사된다. 전혀 다른 repository로 서로의 개발에 전혀 영향을 미치지 않으며 자신이 작성한 코드가 완성되고 기여하고 싶다면 자신의 repository에서 “New pull request”를 통해 요청할 수 있다.

이클립스 개발환경을 사용하는 경우, File -> Import -> Git(Project from Git) -> Clone URI를 선택한다. URI 창에 https://github.com/(your _username)/My SmallBasic을 입력하고 Next를 누른다. Branch Selection 화면에서도 다시 Next를 누른다. 저장할 디렉토리를 설정해준 다음 Next를 누르고, “Import existing Eclipse projects”를 선택하고 넘어간다. 다운로드는 완료되었고, src/com/coducation/smallbasic/gui/MySmallBasicGUI.java를 실행하면 (그림 38)과 같은 마이스몰베이직 프로그래밍 환경이 보여진다. 라이브러리는 src/com/coducation/smallbasic/lib

경로 아래에 추가하고자 하는 라이브러리 클래스를 생성하면 된다. https://github.com/kwanghoon/MySmallBasic에 개발자 가이드를 참고하여 개발을 시작하면 된다.

(그림 38)과 같은 화면에 타이핑을 통해서 스몰베이직 프로그래밍을 할 수 있으며, '열기'를 통해서 기존의 샘플 프로그램들을 열어볼 수 있다. 자신이 작성한 프로그램을 '저장'버튼을 통해 저장할 수 있으며, 기존의 저장된 이름과 다르게 저장하고 싶을 때는 '다른 이름으로 저장'을 누르면 된다. 작성된 프로그램은 '실행' 버튼을 클릭하여 실행할 수 있다.

부록B

제 1 절 마이스몰베이직 개발 상세 업무

표 20 마이스몰베이직 개발 기여도

개발 멤버	담당 업무
최광훈	Lexer, Parser, Interpreter, Library Architecture, GUI, Debugger Arcitecture 설계 및 개발 Desktop 라이브러리 개발 Assert 확장 라이브러리 개발 마이스몰베이직 기반 컴퓨터과학적사고 강의
김가영	BBTransform, PrettyPrinter 개발 Lexical Analyzer, Parser 이슈 해결 및 유지 보수 GraphicsWindow, Controls, Mouse, Shapes, Clock 라이브러리 개발 Chart, Graph, Video 확장 라이브러리 공동 개발 마이스몰베이직 디버거 개발 지원 및 이슈 해결 마이스몰베이직 기반 컴퓨터과학적사고 강의 실습조교
김지용	Text, Program, Math, ImageList, TextWindow 라이브러리 개발 List, Tree, Graph 확장 라이브러리 개발 수업 커리큘럼 개발
박세영	Array, Sound 라이브러리 개발, 마이스몰베이직 GUI 개발 Hamster, Video 확장 라이브러리 개발
정승완	File, Directory, Network, Filckr 라이브러리 개발 Chart 확장 라이브러리 개발
조문영	Tutle, Stack 라이브러리 개발 마이스몰베이직 GUI, 디버거 개발
조성모	Timer, Clock 라이브러리 개발 라이센스 정리 및 Sample, RegressionTest 스몰베이직 프로그램을 활용한 테스팅과 테스트 리포트 작성
김태진	Lexical Analyzer, Parser 개발
김범준	Lexical Analyzer, Parser 개발
조영민	Lexical Analyzer, Parser, AST, Token 개발
HUST(6명)	Database, Facebook 확장 라이브러리 개발

제 2 절 마이스몰베이직 강의에서 사용된 프로그램

2.1 컴퓨터과학적 사고 예제 프로그램: Rush Hour Game

(그림 39)의 소스코드는 프로그램에서 정보 표현을 어떻게 할 것인지에 대해서 학습한 예제 프로그램으로 정보표현 방식 별로 작성된 Rush Hour Game의 블록정보이다.

```
‘ Car Information of Rush Hour Game
‘ Information Represent 3
i = 0
i = i + 1
rushhour[i]["name"] = "A"
rushhour[i]["color"] = "Linen"
rushhour[i]["x"] = 6
rushhour[i]["y"] = 3
rushhour[i]["length"] = 2
rushhour[i]["direction"] = "V"

i = i + 1
rushhour[i]["name"] = "B"
rushhour[i]["color"] = "Goldenrod"
rushhour[i]["x"] = 1
rushhour[i]["y"] = 5
rushhour[i]["length"] = 2
rushhour[i]["direction"] = "V"

i = i + 1
rushhour[i]["name"] = "C"
rushhour[i]["color"] = "PaleTurquoise"
rushhour[i]["x"] = 3
```

```
rushhour[i]["y"] = 5
rushhour[i]["length"] = 2
rushhour[i]["direction"] = "V"

i = i + 1
rushhour[i]["name"] = "D"
rushhour[i]["color"] = "MistyRose"
rushhour[i]["x"] = 5
rushhour[i]["y"] = 5
rushhour[i]["length"] = 2
rushhour[i]["direction"] = "H"

i = i + 1
rushhour[i]["name"] = "E"
rushhour[i]["color"] = "Purple"
rushhour[i]["x"] = 5
rushhour[i]["y"] = 6
rushhour[i]["length"] = 2
rushhour[i]["direction"] = "H"

i = i + 1
rushhour[i]["name"] = "O"
rushhour[i]["color"] = "Yellow"
rushhour[i]["x"] = 3
rushhour[i]["y"] = 1
rushhour[i]["length"] = 3
rushhour[i]["direction"] = "H"

i = i + 1
```

```
rushhour[i]["name"] = "P"
rushhour[i]["color"] = "PapayaWhip"
rushhour[i]["x"] = 1
rushhour[i]["y"] =2
rushhour[i]["length"] = 3
rushhour[i]["direction"] = "H"

i = i + 1
rushhour[i]["name"] = "Q"
rushhour[i]["color"] = "SkyBlue"
rushhour[i]["x"] =5
rushhour[i]["y"] = 2
rushhour[i]["length"] = 3
rushhour[i]["direction"] = "V"

i = i + 1
rushhour[i]["name"] = "R"
rushhour[i]["color"] = "Green"
rushhour[i]["x"] =1
rushhour[i]["y"] = 4
rushhour[i]["length"] = 3
rushhour[i]["direction"] = "H"

i = i + 1
rushhour[i]["name"] = "X"
rushhour[i]["color"] = "Red"
rushhour[i]["x"] =1
rushhour[i]["y"] = 3
rushhour[i]["length"] = 2
```

```
rushhour[i]["direction"] = "H"
```

그림 39 컴퓨터과학적사고 프로그램: Rush Hour Game

2.2 컴퓨터과학적 사고 예제 프로그램: Finding Subway Directions

(그림 40)의 소스코드는 다익스트라 알고리즘으로, 자료구조 그래프에서의 가장 짧은 경로를 찾는 절차를 프로그램으로 작성할 수 있는지를 확인하기 위해 사용되었다.

```
Sub dijkstra
  For i=1 to numberOfNodes
    map[i]["cost"] = inf
    map[i]["prev"] = -1
    map[i]["pattern"] = "False"

    For j=1 to map[i]["count"]
      toNode = map[i][j]
      distx = map[i]["x"] - map[toNode]["x"]
      disty = map[i]["y"] - map[toNode]["y"]
      map[i]["edge"][j] =
                  Math.SquareRoot(distx * distx + disty * disty)

      map[i]["edge"][j] = map[i]["edge"][j] + 10

      If map[i]["line"] <> map[toNode]["line"] Then
        map[i]["edge"][j] = map[i]["edge"][j] + 20
      EndIf
    EndFor
  EndFor

  map[startNode]["cost"] = 0
```

```
While "True"
  u = -1
  For i = 1 to numberOfNodes
    If map[i]["pattern"] = "False" Then
      If u = -1 Then
        u = i
      ElseIf map[u]["cost"] > map[i]["cost"] Then
        u = i
      EndIf
    EndIf
  EndFor

  If u = -1 Then     ' 더 이상 분기할 노드가 없을 때
    Goto ExitDijkstra
  EndIf

  For j = 1 to map[u]["count"]
    w = map[u][j]
    ' 최소의 비용을 가질 때 업데이트
    If map[u]["cost"] + map[u]["edge"][j] <
                                          map[w]["cost"] Then
      map[w]["cost"] = map[u]["cost"] + map[u]["edge"][j]
      map[w]["prev"] = u
    EndIf
  EndFor

  map[u]["pattern"] = "True"
EndWhile
```

```
ExitDijkstra:
  TextWindow.Write (map[startNode]["line"] + " " +
                                  map[startNode]["name"] + "~")
  TextWindow.Write (map[endNode]["line"] + " " +
                                          map[endNode]["name"])

  If map[endNode]["cost"] <> inf Then    ' 도달할 수 있는 경우
    TextWindow.Write(" (" + Math.Round(map[endNode]["cost"])
                                                  + ") : ")
    u = endNode

    While u <> startNode
      Stack.PushValue("path", u)
      u = map[u]["prev"]
    EndWhile

    Stack.PushValue("path", u)
    u = map[u]["prev"]

    pathLen = 0

    While Stack.GetCount("path") > 1
      u = Stack.PopValue("path")

      pathLen = pathLen + 1
      minPath[pathLen] = u

      TextWindow.Write(map[u]["name"] + " -> ")
```

```
    EndWhile
    u = Stack.PopValue("path")

    pathLen = pathLen + 1
    minPath[pathLen] = u

    TextWindow.WriteLine(map[u]["name"])

    Timer.Interval = 10
    Timer.Tick = OnTimer
    timerSeq = 1
    Timer.Resume()
  Else              ' 도달할 수 없는 경우
    TextWindow.WriteLine(" (inf) : unreachable")
  EndIf
EndSub
```

그림 40 컴퓨터과학적사고 예제 프로그램: Finding Subway Directions