프로그래밍 교육 지원을 위한 Unity기반의 GUI 디버깅 도우미

박세찬O, 김덕엽*, 서강복*, 이우진*
O경북대학교 컴퓨터학부,
*경북대학교 컴퓨터학부
e-mail: hasmi5452@gmail.comO, ejrduq77@naver.com*, dating1227@gmail.com*, woojin@knu.ac.kr*

A Unity-Based GUI Debugging Assistant For Programming Education Support

Se-Chan ParkO, Deok-Yeop Kim*, Kang-Bok Seo*, Woo-Jin Lee*
OSchool of Computer Science and Engineering, Kyungpook National University,
*School of Computer Science and Engineering, Kyungpook National University

● 요 약 ●

최근 한국을 포함한 여러 국가들에서 프로그래밍 교육이 중요시 되고 있다. 그러나 이런 상황으로 인해 더 많아진 학생들은 미숙함으로 인해 숙련자에 비해 더 많은 오류를 만나지만 이를 해결하기 위한 디버깅 실력은 아직 미숙하다. 따라서 본 논문에서는 프로그래밍 교육 지원을 위한 초보자용 GUI 디버깅 도우미 UDB (Unity-DeBugger)를 제안한다. UDB는 제출한 학생 코드를 분석하여 반응형 추적표와 오류 로그를 생성하고 이를 기반으로 GUI 및 애니메이션으로 만들어 학생과 상호작용한다. 특히 UDB는 반응형 추적표를 통해 프로그램 안에 있는 변수들의 현재 상태를 보여주고 순방향 추적뿐만 아니라 기존 IDE의 디버깅 도구들과는 다르게 역방향 추적이 가능하다는 큰 특징이 있다. 이런 UDB를 예시 코드에 실제로 적용한 결과를 통해 미숙한 학생도 역방향 추적 기능을 사용하여 오류 원인을 쉽게 찾을 수 있음을 보인다.

키워드: 반응형 추적표(interactive trace table), 역방향 추적(backward tracking), 유니티(Unity)

I. 서론

최근 한국에서는 SW 의무교육이 제도화되었다. 또한 미국과 인도 등의 나라들에서도 프로그래밍 교육이 이루어지고 있다[1]. 이런 교육을 통해 처음 프로그래밍을 배우는 학생들은 낮은 숙련도로 인해 더 많은 오류를 만난다. 그러나 이런 학생들은 스스로 오류를 해결하기 위한 디버깅 실력이 충분하지 않다. 많은 연구에서 학생들은 버그를 찾는 것이 수정하는 것보다 어렵다고 보고하며 예상하지 못한 다양한 디버깅 전략을 쓴다고 보고하였다[2]. 또 기존 디버깅 도구들로 오류를 찾기 위해서는 학생이 미리 오류가 발생하는 지점을 어느 정도 예상할 수 있어야 하고 중단점과 프로시저와 같은 개념을 이해하며 오류와 관련된 변수들을 어느 정도 인지하고 있어야 되기 때문에 처음 프로그래밍을 배우는 학생들에게 기존 디버깅 도구들로 오류를 찾는 것은 다소 어렵다.

따라서 이런 미숙한 학생들이 오류를 찾고 스스로 해결하는 것을 돕기 위해 본 논문에서는 초보자용 GUI 디버깅 도우미 UDB(Unity-DeBugger)를 제안한다. 특히 UDB는 반응형 추적표를 통해 학생과 상호작용하면서 프로그램 안에 있는 변수들의 현재 상태와 그 변화를 요약 및 정렬하여 보여주고, 순방향 추적뿐만 아니라 Visual Studio나 Eclipse와 같은 기존 IDE에 내장된 디버깅 도구들에서는 지원하지 않는 역방향 추적 기능을 제공하다는 큰 특징이 있다.

본 논문의 2장에서는 관련 연구에 대하여 서술한다. 3장에서는 UDB의 개요와 구성, 각 구성 요소의 구현 사항에 대하여 구체적으로 서술하고 UDB 실제 적용 결과를 통해 미숙한 학생도 쉽게 오류 원인을 찾을 수 있음을 보인다. 마지막으로 4장에서는 UDB의 요약, 확장 및 향후 연구방향에 대하여 서술한다.

II. 관련 연구

1. 학생에게 친숙치 않은 추적표를 사용한 pgtracer

pgtracer[3]는 추적표와 프로그램 코드로 이루어진 빈칸 채우기 문제에 대한 프로그래밍 교육 지원 도구이다. 이는 프로그램의 코드에 있는 빈칸을 채우면 자동으로 해당 라인까지의 추적표가 채워지고 이를 정답과 비교하여 학생들에게 보여줌으로써 프로그래밍에 대한 학생들의 이해를 돕는다. 하지만 해당 연구에서는 학생들이 추적표에

친숙하지 않다는 점이 학생들의 이해에 가장 큰 장애물이 된다고 서술했다. 반면에 UDB는 반응형 추적표를 사용하여 학생과 상호작용하며 현재 실행의 변수 값뿐 아니라 이전 실행과 비교하여 값이 변화한 변수들이 앞에 오도록 정렬하고 전체 코드에서 특정 변수의 값 변화를 요약하여 보여준다. 이를 통해 기존 추적표에 친숙하지 않은 학생들도 효과적으로 학습할 수 있다는 장점이 있다.

2. 역방향 추적을 지원하지 않는 기존 디버거들

2022년 기준으로 Visual Studio[4]나 Eclipse[5]와 같은 IDE에 내장된 디버거는 역방향 추적을 지원하지 않는다. 이 때문에 디버깅 중에 오류 발생 라인을 직접 찾은 후 다시 처음부터 디버깅을 시작하여 순방향 추적하며 오류의 원인을 찾아야 된다는 단점이 있다. 반면에 UDB는 오류 발생 라인을 자동으로 찾아주며 역방향 추적을 통해 이전 실행과 오류 발생 시 변수 값들을 되짚어 확인해가며 효과적으로 오류 원인 라인을 찾아볼 수 있다는 장점이 있다.

III. UDB 구현 및 적용 예시

3.1 UDB 개요 및 구성

UDB는 프로그래밍 교육을 위한 초보자용 GUI 디버깅 도우미이다. UDB는 크게 양방향 추적기(Bidirectional Tracer)와 반응형 렌더러(Interactive Renderer)로 구성된다. 양방향 추적기는 Node.js와 GDB를 활용해서 학생 코드에서 반응형 추적표와 오류 로그 등의 데이터를 추출한다. 반응형 렌더러는 Unity를 활용해서 학생 코드, 반응형 추적표, 오류 로그를 GUI 및 애니메이션으로 랜더링한다. 학생은 GUI 및 애니메이션과 상호작용하면서 자신이 제출한 코드의 실행 흐름을 이해하고 오류 발생라인을 찾고 역방향 추적 기능을 사용하여 오류 원인을 효과적으로 찾고 해결할 수 있다. UDB의 전체 구성 요소와 구성 요소 간의 관계는 그림 1에 나타낸다.

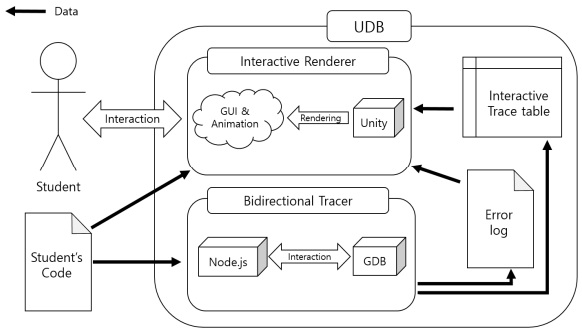

Fig. 1. UDB 시스템 구성도

3.2 양방향 추적기 구현과 반응형 추적표 생성

양방향 추적기는 Node.js와 GDB를 활용하여 학생 코드에서 반응형 추적표와 오류 로그를 추출하는 UDB의 구성요소이다. 먼저 Node.js에서 GDB를 구동하는 자식 프로세스를 생성해 부모 프로세스와 pipe로 소통하며 GDB 명령들을 실행함으로써 정보를 추출하여 반응형 추적표를 생성한다. 이렇게 미리 디버깅하여 생성된 반응형 추적표가 UDB의 역방향 추적을 가능하게 한다.

반응형 추적표 생성 과정은 다음과 같다. 먼저, GCC로 코드를 컴파일해 목적파일을 생성하고 "gdb <목적파일명>"명령으로 GDB 프로그램을 시작한다. 이후 프로그램이 바로 종료되는 걸 막기 위해 "b 1"명령으로 브레이크 포인트를 설정한다. 그 다음으로 "run"명령으로 디버깅을 시작한다. 이제 프로그램이 종료될 때까지 "gdb next"와 표 1의 명령어를 실행하여 반응형 추적표 생성에 필요한 정보를 추출하고 이를 기록한다. 프로그램이 종료되거나 오류로 인해 더 이상 실행할 수 없다면 "gdb quit"를 통해 디버깅을 종료하고 기록한 정보들을 엑셀 형식으로 변환하여 반응형 추적표를 생성한다. 여기서 생성된 반응형 추적표는 실행 번호, 라인 번호, 라인이 속한 함수, 각 변수의 값, 출력 값으로 구성된다. 각 정보를 얻기 위한 GDB 명령어는 표 1에 나타낸다.

Table 1. 반응형 추적표의 정보를 얻기 위한 GDB 명령어

정보 종류	필요한 GDB 명령어
실행 번호	node.js에서 cnt
라인 번호	gdb where
소속 함수 이름	gdb frame
각 변수의 값	gdb variables info
출력값	없음(gdb next시 출력값)

3.3 반응형 렌더러 구현과 화면 구성

반응형 렌더러는 학생코드 및 양방향 추적기가 추출한 반응형 추적표, 오류 로그를 GUI 및 애니메이션으로 랜더링하는 UDB의 구성요소이다. 이는 Unity 게임 엔진으로 구동된다.

반응형 렌더러로 생성한 UDB의 화면은 크게 4가지 영역으로 구성된다. 좌측 상단의 코드 영역, 좌측 하단의 반응형 추적표 영역, 우측의 애니메이션 영역, 최하단의 유저 입력 영역이다. 그림 2는 전체화면과 각 영역의 매긴 번호를 나타낸다.

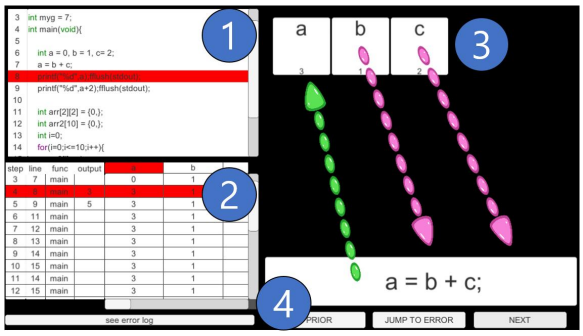

Fig. 2. UDB 전체 화면과 각 영역

1로 표시된 코드 영역은 학생이 제출한 코드를 보여주며, 다음에 실행할 라인을 붉은 색으로 강조한다. 또 학생의 편의를 위해 구문강조를 통해 각 키워드나 문법에 해당하는 단어에 적절한 색을 입힌다. 그림 3은 코드 영역만을 확대한 화면을 나타낸다.

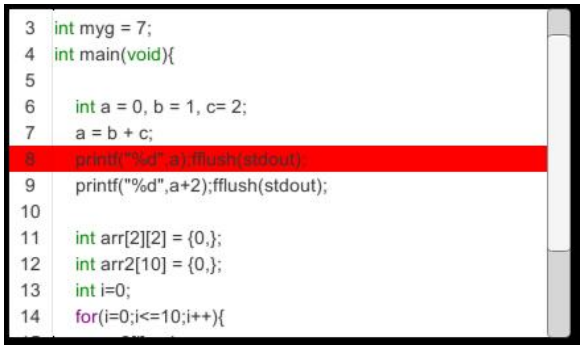

Fig. 3. 코드 영역 확대 화면

2로 표시된 반응형 추적표 영역은 프로그램의 각 라인이 실행될 때마다 기록된 현재 실행 번호, 다음 실행 라인 번호, 다음 실행의 출력값, 현재 소속 함수 이름, 변수들의 현재 값들 등을 보여준다. 또 현재 라인의 값들은 붉은 색으로 강조되며 이전 실행과 비교하여 변화된 변수들의 이름도 붉은 색으로 강조되며 변화된 변수들이 앞으로 오도록 정렬한다. 이를 통해 미숙한 학생에게도 한눈에 변화를 파악시키고 오류의 원인 쉽게 찾도록 도와줄 수 있다. 그림 4는 반응형 추적표 영역만을 확대한 화면을 나타낸다.

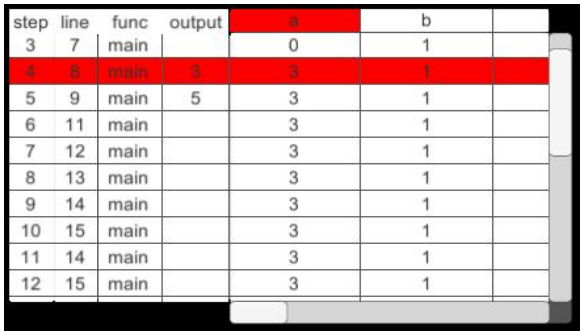

Fig. 4. 반응형 추적표 영역 확대 화면

3으로 표시된 애니메이션 영역은 현재 실행 라인이 실행되는 과정을 해당 라인의 코드와 화살표, 변수 등을 통해 애니메이션으로 보여준다. 붉은 색 화살표는 사각형 흰색 박스로 표시된 메모리에 있는 변수가 코드 실행 시 참조되는 것을 나타내고, 초록색 화살표는 코드 실행에서 계산된 값으로 메모리에 있는 변수를 초기화 하는 것을 나타낸다. 각 화살표는 페이드인 애니메이션이 적용되었다.

4로 표시된 유저 입력 영역은 오류 로그 보기(see error log), 이전 실행으로(prior), 오류로 점프(jump to error), 다음 실행으로(next)의 4개의 버튼으로 구성되어 있다. 오류 로그 보기는 GDB에서 stderr로 출력된 오류 로그를 볼 수 있다. 이전 실행으로 버튼은 코드 영역과 반응형 추적표 영역의 붉게 강조된 영역을 이전 실행 라인으로 바꾸고 이전 실행의 애니메이션을 보여준다. 오류로 점프 버튼은 만약 오류가 존재한다면 해당 오류 발생 라인으로 점프하고 그렇지 않다면 오류가 없다는 메시지를 팝업으로 출력한다. 다음 실행으로 버튼은 코드 영역과 반응형 추적표 영역의 붉게 강조된 영역을 다음 실행 라인으로 바꾸고 다음 실행의 애니메이션을 보여준다. 그림 5는 유저 입력 영역만을 확대한 화면을 나타낸다.

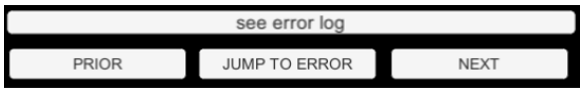

Fig. 5. 유저 입력 영역 확대 화면

또 반응형 추적표 영역의 상단에 위치한 각 변수의 이름이나 함수, 출력값 헤더를 클릭하면 해당하는 값의 변화를 요약한 표가 팝업으로 나타난다. 표에는 해당 값의 변화할 때만의 실행 번호와 라인 번호, 값 등을 보여주어 해당 값의 변화를 한눈에 보고 쉽게 오류 원인을 찾도록 돕는다. 또 편의를 위해 코드 영역과 반응형 추적표 영역에서 값을 클릭하면 팝업을 통해 값을 큰 화면에서 확인할 수 있도록 하였다. 본 논문에서 구현한 양방향 추적기와 반응형 렌더러에 대한 소스코드는 깃허브[6][7]에 공개되어있다.

3.4 UDB 적용 실제 사례와 역방향 추적

본 절에서는 UDB의 반응형 추적표와 역방향 추적을 통해 학생이 쉽게 오류 원인을 찾을 수 있음을 보이기 위해 예시 코드를 UDB에 적용한 사례를 보인다. 예시 코드는 미숙한 학생들이 자주 만날 수 있는 인덱스 범위를 벗어나는 오류가 발생하는 코드이다.

먼저 UDB를 실행하고 오류로 점프 버튼을 누르면 오류 발생 라인 화면과 오류라는 경고 메시지가 출력된다. 오류 발생 화면은 그림 7에 나타낸다. 오류 발생 화면을 보면 오류 발생 라인에서 arr2[i]에 접근할 수 없어서 “??”가 애니메이션 영역에 나타났다.

Fig. 7. 오류 발생 라인 화면

이에 오류의 원인을 알기 위해 이전 실행으로 버튼을 누르면 이전 실행에 대한 화면이 나타난다. 이전 실행 라인 화면은 그림 8에 나타낸다. 이전 실행 화면을 보면 반응형 추적표와 애니메이션에서 i라는 변수의 값이 10임을 알 수 있다. 하지만 이전에 arr2 배열은 크기가 10으로 할당되었음으로 i가 인덱스 범위인 0~9를 벗어난 것이 오류 원인임 알 수 있다.

이 사례를 통해 UDB를 활용하면 자동으로 오류 발생 라인을 찾아주고 해당 라인에서의 변수 정보 등을 볼 수 있으며 역방향 추적을 통해 효과적으로 오류 원인을 찾을 수 있음을 알 수 있다.

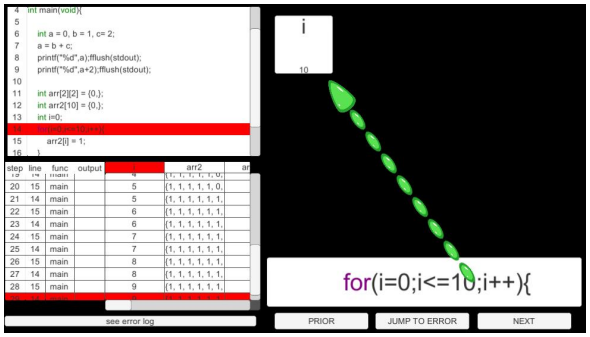

Fig. 8. 이전 실행 라인(오류 원인 라인) 화면

IV. 결론

본 논문에서는 프로그래밍 교육을 위한 초보자용 GUI 디버깅 도우미 UDB를 구현하였다. UDB는 Node.js와 GDB를 사용하여 학생 코드에서 반응형 추적표, 오류 로그를 추출하며 Unity를 활용하여 학생 코드와 반응형 추적표, 오류 로그를 GUI 및 애니메이션으로 랜더링한다. 특히 UDB는 미숙한 학생이 디버깅에 대한 이해가 부족하더라도 자동으로 오류를 찾아주며 오류 발생 라인에서 멈춰주며 기존 디버깅 도구들에는 없는 역방향 추적 기능을 통해 오류 지점부터 되짚어 가며 오류의 원인이 되는 코드를 효과적으로 탐색할 수 있다. 또한 예시 코드에 UDB를 적용한 실제 사례를 보여줌으로써 미숙한 학생도 오류 원인을 효과적으로 찾을 수 있음을 보였다.

향후에는 본 논문에서 제안한 UDB가 더욱 다양한 언어를 지원하고 웹으로 제공하도록 개선하여 접근성과 범용성을 확보해야 할 것이다. 또 웹을 통한 학생들의 GDB 사용 기록 수집을 통해 프로그래밍 교육을 위한 새롭고 유의미한 데이터를 확보할 수 있을 것이다. 특히 이식성이 높은 Unity를 사용하여 구현하였기 때문에 웹으로 확장이 용이할 것으로 기대된다.

ACKNOWLEDGEMENT

이 논문은 2018년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업(No. NRF-2018R1A6A1A03025109)이며 교육부 및 한국연구재단의 4단계 BK21 사업(경북대학교 컴퓨터학부 지능융합 소프트웨어 교육연구단)으로 지원된 연구임(4199990214394).

REFERENCES

[1] Eunkyoung Lee. "Perspectives and Challenges of Informatics Education: Suggestions for the Informatics Curriculum Revision," The Journal of Korean Association of Computer Education Vol. 21, No. 2, pp. 1-10, March 2018.

[2] S. Fitzgerald, R. McCauley, B. Hanks, L. Murphy, B. Simon and C. Zander, "Debugging From the Student Perspective," IEEE Transactions on Education, Vol. 53, No. 3, pp. 390-396, Aug. 2010, doi: 10.1109/TE.2009.2025266.

[3] T. Kakeshita and M. Murata, "Utilizing Programming Education Support Tool Pgtracer in an Actual Programming Course," 2017 6th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI), 2017, pp. 471-476, doi: 10.1109/IIAI-AAI.2017.36.

[4] https://docs.microsoft.com/ko-kr/visualstudio/debugger/?view=vs-2022

[5] https://help.eclipse.org/latest/index.jsp?topic=/org.eclipse.jdt.doc.user/reference/views/debug/ref-debug_view.htm

[6] https://github.com/amkorousagi/trace_table_generator

[7] https://github.com/amkorousagi/udb