Journal of The Korean Association
of Information Education Vol. 20, No. 5,
October 2016, pp. 519-526

http://dx.doi.org/10.14352/jkaie.2016.20.5.519
© 2016 KAIE

Computational Thinking 향상 도구로서의 App Inventor 2

임병춘
춘천교육대학교 컴퓨터교육과

요 약

MIT의 App Inventor 2는 스크래치나 Alice와 같은 블록-기반 프로그래밍 언어로 학생들이 CT의 개념과 기능을 갖추도록 하는데 있어 매우 유용하게 사용될 수 있는 도구이다. 스크래치 프로그래밍은 프로그램 작성과 실행이 대부분 컴퓨터상에서 이루어지는 반면에 App Inventor 2는 컴퓨터상에서 프로그램을 작성하는 중간에 스마트폰으로 그때까지의 결과를 직접 확인하고 즉각적인 피드백을 받으면서 프로그래밍을 하기 때문에 더 좋은 학습 효과를 기대할 수 있다. 이 연구에서는 학생들의 App Inventor 2 학습이 학생들의 CT 기능 향상에 기여하는지를 규명하기 위해 CT 개념 중 네 가지 핵심 개념을 추출하여 App Inventor 2 학습에서 이 네 가지 개념이 어떻게 구현되는지를 보인다.

키워드 : Computational Thinking, 앱 인벤터 2, 모바일 프로그래밍, 앱 프로그래밍, 코딩 교육

App Inventor 2 As a Tool for Enhancement of Computational Thinking

Byeongchoon Lim
Chuncheon National University of Eduction

ABSTRACT

MIT App Inventor 2, much like the other block-based programming languages such as Scratch and Alice, is a useful tool that can be utilized to improve the computational thinking skills for students. The coding and the execution of program are done mainly on the computer in Scratch programming, but in App Inventor 2 programming, the students can execute the program and get immediate feedback on the smart-phone in the middle of the programming. Therefore, we can expect better enhanced learning effect. In this study, it has been claimed that the learning of App Inventor2 can strengthen the computational thinking skills of our students by extracting four core concepts of the computational thinking and then showing how these concepts are applied through the learning of App Inventor 2.

Keywords : Computational Thinking, App Inventor 2, Mobile Computing, App Programming, Coding Education

논문투고 : 2016-10-01
논문심사 : 2016-10-04
심사완료 : 2016-10-26

1. 서론

컴퓨터 프로그래밍 경험이 없는 사람들도 과거처럼 복잡한 코드를 작성하지 않고도 단지 퍼즐 조립하듯이 블록을 조립하는 방식으로 쉽게 프로그래밍할 수 있도록 하는 Scratch, Alice 등과 같은 프로그래밍 도구들이 등장한 이후 컴퓨터과학을 전공하지 않는 사람들도 큰 어려움 없이 프로그래밍을 할 수 있는 여건이 갖추어졌다. 이와 함께 우리나라와 미국은 물론 세계 많은 나라에서 코딩교육을 강화하는, 특히 어렸을 때부터 코딩교육을 시행하기 위한 정책들을 잇달아 내놓으면서 코딩교육을 통해 교육의 효율성과 효과를 향상시키기 위한 연구가 활발히 진행되고 있다.

Code.org(https://code.org) 누리집의 첫 번째 페이지에 있는 동영상(Why Coding?)에서 애플의 스티브 잡스는 “이 나라의 모든 사람들은 컴퓨터를 프로그램하는 방법을 배워야 한다. (중략) 왜냐하면 이것이 생각하는 방법을 가르쳐주기 때문이다.”, 마이크로 소프트의 빌 게이츠는 “13 살 때 나는 처음으로 컴퓨터를 접했다.”, 트위터의 잭 도시는 “나의 부모님은 내가 8 살 때인 1984년에 매킨토시를 사주었다.”, 그리고 페이스북의 마크 저커버그는 “나는 6학년 이었다.”고 말한다. 이와 같이 이름만 대면 누구든지 아는 전 세계적으로 크게 성공한 사람들의 사례를 접하는 학부모들은 아이들의 조기 코딩교육의 필요성을 실감할 것이다. 한편 미 대통령 오바마는 2016년 일반 교서에서 다음과 같이 학생들에게 직접 해보는 컴퓨터과학을 제공하여야 한다고 강력하게 주장하였다.

> In the coming years, we should build on that progress, by … offering every student the hands-on computer science and math classes that make them job-ready on day one.
>
> - Computer Science For Alla
> (a blog in the WhiteHouse site)

Computer Science for All에서는 보다 구체적으로 유치원에서부터 고등학교까지 미국의 모든 학생들에게 컴퓨터과학을 배우게 하고 Computational Thinking(이하 CT) 능력을 갖추도록 한다고 함으로써 조기 코딩교육의 시행을 강조하였다. 사실 조기 코딩교육을 최초로 실시한 국가는 영국으로 2014년 9월부터 모든 학년에서 의무적으로 컴퓨터 프로그래밍을 학습하도록 하였다. 우리나라의 경우 초등 코딩교육과 관련하여 2015년 9월에 교육부에서 발표한 문이과 통합 교육과정에 초등학교 5, 6학년을 대상으로 하는 소프트웨어 교육 강화 내용이 포함되었고 2017년부터 실시된다는 내용이 포함되었다[14]. 교육부의 발표 이후 서울의 강남을 중심으로 어린 아동들을 대상으로 하는 코딩교육 열풍이 일었으며 동시에 이의 부작용을 우려하는 사회적 분위기도 조성되었다.

> 코딩, 쉽게 말해 컴퓨터 언어를 가리키는 단어입니다. 최근 미국 오바마 대통령이 (중략) 교육부가 2018년부터 (중략) 그 사실이 알려지자마자 역시나 선행학습 시장이 들끓고 있습니다. 월 2백만 원짜리 코딩 유치원, 그리고 8백만 원짜리 코딩 캠프까지 등장했습니다. (중략) “코딩교육을 공교육에서 잘하면 얼마나 좋겠습니까. 교육들 철저히 해줄 수 있는 분들의 숫자가 많이 적다는 게 현실입니다.”(김진형 소장/소프트웨어 정책연구소) JTBC 카드 뉴스

코딩교육을 담당할 교사가 많이 부족하다는 지적은 초등학교 교사들의 견해와도 일치한다. 교사를 대상으로 하여 매우 많은 코딩과 ICT 관련 연수 프로그램이 다양하게 제공되고 있으나 매번 같은 교사가 교육에 참여하여 실제 교육을 받는 교사의 수는 매우 적을 뿐만 아니라 심지어 코딩교육을 담당할 수 있는 교사가 한 명도 없는 초등학교가 매우 많은 것이 현재의 실정이다. 이러한 문제를 가장 빠르게 해결하는 방법으로 현직 교사들을 대상으로 하는 코딩교육 연수를 크게 활성화하고 연수자 중에서 일정 수준 이상의 능력을 갖춘 교사들에게 인센티브를 제공하는 등의 정책을 적극적으로 시행하는 것이 있고, 장기적으로는 초등교사를 양성하는 교육대학의 교육과정에서 ICT 관련 교과의 비중을 높이고 동시에 코딩교육의 내실을 기하는 것이 있다. 현재 대부분의 초등교원 양성 대학에서는 모든 학생이 스크래치 프로그래밍을 의무적으로 수강하도록 되어 있으며 모바일 프로그래밍을 개설하여 제공하고 있는 대학도 있다.

이 연구에서는 App Inventor 2(이하 앱 인벤터)를 사용한 앱 개발 수업에서 실제로 다루고 있는 내용을 분석하여 앱 인벤터 학습이 조기 코딩교육을 담당할 초등학교 예비교사들의 CT 능력 향상에 도움이 되는지에 관하여 분석한다. 이를 위하여 먼저 CT에 관한 현재까지의 연구 동향을 조사하고 CT 요소들 중에서 핵심 요소들을 추출하여 앱 인벤터 학습이 이들과 관련된 기능 향상에 매우 유용한 도구로서 사용될 수 있음을 보인다.

2. CT의 핵심 역량

Wing[25]이 CT와 관련하여 모든 아동들의 논리적 능력에 아동들의 학습에 주요한 요소로서 CT가 추가되어야 한다고 주장한 이후 현재까지 컴퓨터과학이나 교육 분야에서 CT가 매우 뜨거운 논쟁을 일으켰고 또한 수많은 연구가 이루어져왔다. 사실 아동들의 프로그래밍 학습이 아동들의 논리적 사고력을 향상에 도움이 된다는 주장은 이미 1980년에 Papert에 의해서 제기되었다. 많은 연구에서 CT를 프로그래밍과 연관 지어 서술하였기 때문에[10] CT를 프로그래밍과 동일한 것으로 또는 CT는 프로그래밍의 사용을 필요로 하는 것으로 보기도 하였지만 CT가 컴퓨터과학으로만 한정되지 않고 모든 학문에 걸쳐 적용 가능하며, CT 능력 향상을 위하여 반드시 프로그래밍의 사용이 필요한 것은 아니라는 것이 최근의 추세이다. 다만 오늘날 대부분의 직업에서 그리고 거의 모든 사람들의 일상에서 컴퓨터의 사용을 필요로 하고 있기 때문에 CT 능력 향상 수단으로 컴퓨터과학 특히 프로그래밍 학습을 통하는 것이 자연스럽고 보다 효과적일 것이다.

다양한 연구자들과 단체들이 초·중·고등학교에서의 CT를 정의하였다[9]. Google for Education의 CT 개요[12]에서는 CT를 구성하는 개념들로 추상화, 알고리즘 설계, 자동화, 데이터 분석, 자료 수집, 분해, 자료 표현, 병렬화, 패턴 일반화, 패턴 인식과 시뮬레이션을 들었다.

한편 Barr 등[3]은 CT 개념과 역량의 학습을 지원하고 강화하는 일련의 자질과 태도로 복잡한 문제를 다루는데 있어서의 자신감, 어려운 문제를 해결하는데 있어서의 끈기, 모호함을 다루는 능력, 해가 하나만 있는 것이 아닌 개방형 문제를 다루는 능력 등이 있음을 강조하였다.

많은 연구와 문헌에서 CT에 대하여 각기 다른 여러 가지 다양한 정의가 제시되어 왔는데 대부분의 연구에 공통적으로 포함되는 CT의 개념 또는 역량으로는 분해, 패턴 인식, 추상화, 알고리즘이다. 구글의 Computational Thinking for Educators[11]는 구글에서 교사들의 CT 역량 강화를 위해 제공하는 온라인 코스로 구글이 제시한 CT의 개념 중 이 네 가지 개념을 중심으로 하는 내용을 담고 있다. BBC BiteSize[4]에서도 이 4 가지 핵심 기술에 관하여 집중적으로 설명하고 있다.

이 연구에서는 간단한 앱 인벤터 프로젝트를 분석하여 앱 인벤터 학습이 학생들의 CT 핵심 역량의 향상에 직접적인 영향을 끼칠 수 있다는 것을 보인다.

3. CT 역량 강화를 위한 앱인벤터 학습

아이들은 어려서부터 수많은 추상적 개념을 배운다. 그런데 Piaget는 아이들은 형식적 조작기(14세 이후)에 들어서서야 추상적 방식으로 생각한다고 하였다[1]. 이는 조기 코딩교육이 효과적이지 않을 수도 있음을 시사하는데 이에 반하여 Papert21]는 생각할 사물을 직접 갖고 학습하는 것은 추상적 개념의 이해를 증진시킨다고 하였다. 따라서 자신이 작성한 프로그램을 즉각적으로 확인할 수 있다는 것은 학생들이 즉각적인 피드백을 받을 수 있다는 것을 의미하며 이것은 학생들이 추상적 개념과 아이디어들을 명확하고 확실한 방법으로 이해하는데 분명히 도움이 된다.

Papert는 이미 오래 전에 프로그래밍 학습이 아동들의 논리적 사고력 향상에 도움이 된다고 주장하였으나 실제로 교육 현장에서 아동들을 대상으로 한 코딩교육은, 프로그래밍은 어려운 것이라는 생각 그리고 물리적 환경이 갖추어지지 않은 등의 이유도 있지만 무엇보다도 코딩교육이 단순히 아동들의 컴퓨터 조작 능력을 기를 뿐이라는 인식에 활성화되지 못하였다. 그러나 스마트폰 없는 아이들이 없고, 과거와 같이 문장을 작성하듯이 프로그램을 작성하지 않고 마치 퍼즐을 맞추듯이 쉽게 프로그래밍 할 수 있는 도구들이 많이 등장하였고, 이들 도구를 사용하여 프로그래밍하는 사람들이 폭발적으로 많아지면서 최근에 들어서야 조기 코딩교육에 대

한 사회적 관심이 높아지게 되었다. 이러한 현상에 이론적 기반을 마련한 것이 바로 CT 연구라고 할 수 있다.

Wing[25]의 CT에 관한 연구 이후로 현재까지 이와 관련된 연구가 무척 활발하게 이루어져 왔다. CT를 비단 컴퓨터과학에 뿐만 아니라 교육 전반에 도입하는 것과 관련된 연구가 상당히 많이 이루어졌는데 반해 프로그래밍이 CT 역량 강화에 도움이 되는가?에 관한 연구는 상대적으로 적게 이루어져왔다. 최근에 들어서 스크래치 사용자가 비약적으로 늘어나면서 스크래치 프로그래밍이 학생들의 문제 해결 능력이나 CT 역량을 향상시킨다고 주장하는 국내의 연구들이 이루어졌다[2][5][7][16][17]. 특히, 가장 최근의 연구로 최형신[7]은 CT의 세부 역량의 개발을 위해 교원양성기관에서 한 학기동안 진행할 수 있는 수업을 설계하고 각 차시의 수업 내용과 CT 요소와의 관련성을 분석하여 제시하였고 또한 PECT 모형과 CT 평가 프레임워크를 분석하여 학습자의 산출물에 나타난 CT 역량 습득 정도를 평가할 수 있는 루브릭을 고안하였다.

이와 같이 CT 역량 향상과 관련하여 스크래치 프로그래밍에 관한 연구는 비교적 활발히 이루어졌으나 앱 인벤터와 관련된 국내의 연구는 거의 이루어지지 않았다. 그것은 스크래치와 앱 인벤터가 모두 MIT 미디어랩에서 유지 관리되고 있지만 앱 인벤터가 스크래치보다 훨씬 늦게 배포되기 시작된 것이 이유 중 하나일 것으로 판단된다. 해외의 경우 Mobile Computing(Mit App Inventgor) Mobile Computational Thinking[24])와 같은 용어를 사용하여 앱 인벤터를 사용한 CT 역량 강화와 관련된 문헌들이 등장하기 시작했다.

스크래치와 앱 인벤터 모두 프로그래밍 경험이 없은 사람들도 코드 블록들을 드래그 앤 드롭하여 쉽게 원하는 프로그램을 완성하도록 하는 블록-기반 프로그래밍 환경이며 코드 작성 방법 또한 매우 유사하다. 스크래치에 센서 보드를 사용하는 기능이 있기는 하지만 스크래치 프로그래밍은 주로 컴퓨터 상에서 이루어지고 완성된 프로그램의 실행도 주로 컴퓨터에서 진행되는 반면에 앱 인벤터는 핸드폰의 사용을 전제로 그리고 실제로 프로그램의 실행을 그때그때 핸드폰을 손에 들고 확인하면서 코딩이 진행되기 때문에 Papert의 주장대로 학생들이 자신이 코딩한 것에 대하여 즉각적인 피드백을 받을 수 있고 이것은 학생들이 추상적 개념과 아이디어들을 명확하고 확실한 방법으로 이해하는데 더 효과적일 것이다.

앱 인벤터를 사용하여 앱을 개발할 때에는 과거와 같이 문장을 일일이 입력하여 코딩하는 방식이 아니라 앱 화면에 포함될 개체들을 원하는 위치에 끌어다 놓는 방식으로 시각적으로 설계하고, 앱의 기능은 각 개체와 관련된 블록들을 조립하는 식으로 코딩을 하기 때문에 기존의 프로그래밍 언어를 사용하는 것보다 훨씬 수월하게 코딩하는 것이 가능하다. 이러한 점에서 앱 인벤터는 스크래치나 Alice와 같은 프로그래밍 언어와 매우 유사하지만 차이점은 앱 인벤터는 궁극적으로 스마트폰에서 사용하는 것을 전제로 프로그래밍한다는 것이다[20].

앱 인벤터는 프로그래밍 언어로서 여러 가지 장점들을 가지고 있다. 코딩 작성을 매우 쉽게 할 수 있음에도 불구하고 기존의 프로그래밍 언어들이 갖고 있는 대부분의 고급 기능을 구현할 수 있고, 앱인벤터를 사용한 모바일 컴퓨팅의 강력함을 지렛대 삼아 교사와 학생들이 스스로 필요한 앱을 설계하고 만들 수 있으며[13], 앱 인벤터로 코딩 학습하는 동안 학생들은 자연스럽게 클라우드 컴퓨팅 환경에 친숙해진다.

4. 앱인벤터 학습을 통한 CT 구현

앞에서 CT의 핵심 역량으로 분해, 패턴 인식, 추상화와 알고리즘을 들었다. 분해는 원래의 문제를 다루기 쉽도록 더 작은 부분들 즉 작은 문제들로 나누는 것이고, 패턴 인식은 문제들 간의 또는 작은 문제들 사이의 유사성를 찾는 것이고, 추상화는 관련성이 적은 상세들을 숨기거나 무시하고 중요한 정보에만 초점을 맞추는 것이고, 알고리즘은 문제를 해결하기 위한 단계적인 절차 또는 문제를 해결하기 위해 따르는 규칙을 만드는 것이다. CT 기능은 일단 문제를 대하면 문제를 다루기 쉬운 작은 문제들로 나누고, 이전에 해결된 문제 중 유사한 문제가 있는지 그리고 분해된 작은 문제들 사이에 유사성이 있는지를 찾아보고, 관련 없는 상세들을 무시하면서 중요한 초점에만 집중하고, 작아진 문제들을 각각 해결하는 단계들이나 규칙들을 설계할 수 있는 능력이라고 할 수 있다..

MIT 앱인벤터 사이트는 앱 인벤터 학습을 위하여 초

급, 중급, 고급으로 등급이 정해진 총 25 개의 프로젝트 튜토리얼을 제공하고 있다. 이들 프로젝트를 모두 또는 그들 중 일부를 체계적으로 선정하여 학습하면 학습자는 탄탄한 CT 역량을 갖게 될 것이다.

이 연구에서는 튜토리얼에 포함된 프로젝트 중에서 가장 난이도가 낮은 것으로 판단되는 HelloPurr for App Inventor 2를 해결하는 과정에서 CT 개념이 적용되는 것을 보임으로써 앱 인벤터 학습이 CT 기능 향상에 도움이 될 수 있음을 규명한다.

(Fig. 1)은 해결하려고 하는 원래의 문제(화면의 고양이 그림이 그려져 있는 버튼을 누르면 고양이 울음소리가 나도록 하는 문제)이다. 버튼의 배경 이미지로 사용할 그림 파일과 고양이 울음소리로 사용할 사운드 파일은 앱 인벤터 서버에 이미 업로드된 것으로 가정한다.

(Fig. 1) HelloPurr Project

분해: 복잡한 문제는 처음 보면 해결하는 방법을 쉽게 알기가 어렵다. 문제를 대했을 때 이를 해결하기 위하여 생기는 의문들이나 처리해야 할 항목들을 나열하면 작은 부분들로 나누기 수월해진다. 일단 학생들은 이 문제를 해결하기 위하여

- 앱에 필요한 개체들은 무엇인가?(Screen1, TextBox, Button, Sound)
- Screen1 개체의 모양(제목, 배경 등)은 어떻게 할 것인가?
- Button 개체의 모양(개체명, 크기, 바탕색, 텍스트, 글꼴, 글꼴 크기 등)은 어떻게 할 것인가?
- TextBox 개체의 모양(개체명, 크기, 배경, 텍스트, 글꼴, 글꼴 크기 등)은 어떻게 할 것인가?
- Sound 개체에 고양이 울음소리 파일 연결은 어떻게 하는가?
- Button 개체를 눌렀을 때 고양이 울음소리가 나게 하기 위해서는 어떻게 해야 하는가?

와 같은 질문을 스스로 할 것이다. 이와 같은 질문의 나열은 원래의 문제를 작은 여러 개의 문제로 나눈 것이라고 할 수 있다. 분해는 전체를 부분들로 나누어 생각하는 능력이다. 원래의 문제를 분해하는 것을 통해서 작고 다루기 쉬운 문제들로 전환될 수 있으며 이들 각각의 작은 문제를 해결하는 과정은 보다 쉽게 개발될 수 있다. 이 과정은 또한 역으로 나중에 통합되어 원래 문제의 해결로 이어진다. 적절하게 부분들로 나뉘었을 때 각 부분을 독립적으로 이해하고 해결하는 것은 전체를 한꺼번에 해결하는 것보다 훨씬 다루기 쉽고 따라서 해를 구하는 것이 수월해지고 나중에 전체를 이해하고 평가하기 쉽게 해준다.

패턴 인식: 문제를 대했을 때 이 문제와 유사한 문제가 이미 존재하는지 그리고 분해된 작은 문제들 사이의 유사성이 존재하는지를 조사하는 것은 중요하다. 패턴에는 문제들 사이의 패턴과 문제 내부의 패턴(작은 문제들 사이의 패턴)이 있다. HelloPurr 프로젝트는 앱 인벤터를 학습하는 사람들이 대개 처음 접하게 되는 프로젝트이므로 HelloPurr 프로젝트와 유사한 앱 인벤터 프로젝트를 경험해본 적이 없기 때문에 문제들 사이의 패턴을 발견하기 어렵다. 그러나 만일 스크래치나 Alice로 유사한 경험이 있다면 그 패턴을 현재의 문제를 해결하는데 유용하게 사용할 수 있을 것이다.

HelloPurr 프로젝트에서 Botton 개체의 크기를 설정하는 것이나 TextBox 개체의 크기를 설정하는 과정은 거의 동일한 패턴으로 문제 내부의 패턴이다. 어떤 한

개체의 속성을 설정하는 방법은 다른 개체의 속성을 설정하는 방법과 유사하다. 예를 들어 Button 개체의 어떤 속성을 설정할 줄 알면 다른 어떤 개체의 속성을 설정하는 것에 대하여 이해하게 되고 그러면 Screen1 개체의 제목 속성이나 Sound 개체의 Source 속성도 쉽게 설정할 수 있게 된다. 이렇게 알게 된 각 개체의 속성을 설정하는 방법은 앞으로 다른 문제를 해결하고자 할 때 문제들 사이의 패턴으로도 활용될 수 있다. 이와 같이 하나의 문제를 해결하는 과정에서 발견된 패턴들은 이후 다른 문제를 해결할 때 유용하게 사용될 수 있는 패턴이 된다.

마찬가지로 Button 개체를 클릭했을 때 고양이 울음소리가 나도록 하는 방법을 알면 Button 이외의 다른 어떤 개체를 클릭했을 때도 그에 대한 처리가 이루어지는 것에 대해 쉽게 이해할 수 있고 다른 문제 해결에 활용될 수 있다.

패턴을 찾는 것은 문제 해결을 단순하게 해주기 때문에 매우 중요한데 그것은 문제들이 패턴을 공유하게 되면 문제 해결이 쉬워지기 때문이다.

추상화: 추상화는 문제 해결에 필요한 상세들을 모아 하나의 문제로 일반화하고 불필요한 상세들은 숨기거나 제거하는 것이다. 패턴 인식이 제대로 이루어지면 유사한 패턴끼리 묶어 일반화가 수월해진다. 예를 들어 6 개의 작은 문제들 중에서 Screen1 개체의 모양, Button 개체의 모양, TextBox 개체의 모양을 정하고 Sound 개체에 소리파일 연결하는 것의 네 개의 작은 문제는 개체의 속성을 설정하는 문제로 일반화가 가능하다. 따라서 6 개의 작은 문제들은

- 필요한 개체 선정
- 각 개체 속성 설정
- Button 클릭 이벤트 처리

와 같이 세 개의 문제로 축소될 수 있다. 만일 HelloPurr 프로젝트에서 Button 이외의 다른 개체를 클릭하는 이벤트가 필요한 경우에는 Button 클릭 이벤트와 이 이벤트를 개체 클릭 이벤트라는 것 하나로 묶을 수 있을 것이다. 실제로 앱 인벤터로 개발하는 모든 앱은 이 세 가지 문제를 해결하는 문제라고 할 수 있다. 따라서 앱 인벤터로 앱을 개발하는 일반적인 과정은

개체 선택 → 개체 속성 설정 → 이벤트 처리

와 같이 정리할 수 있다. 그리고 세 가지 각각의 문제는 다른 프로젝트를 수행할 때 문제들 사이의 패턴으로도 사용이 가능하다.

알고리즘: 알고리즘은 문제를 해결하기 위한 일련의 단계적 지시들로 문제 해결을 위한 일종의 계획이라고 할 수 있다. 학생이 문제를 해결하려고 할 때 이를 위한 계획이 없다면 컴퓨터도 문제를 해결할 수 없기 때문에 문제를 해결하는 상세하고 정확한 절차(순서가 정해진)들이 주어져야만 그에 따른 일을 처리할 수 있다. 알고리즘이 정확하게 수립되지 않으면 문제 해결이 잘못되거나 아예 해결이 되지 않는다. 분해와 패턴 인식이 제대로 되어야만 정확한 알고리즘을 수립할 수 있으며 알고리즘을 순서도로 그리거나 의사코드로 작성하면 코딩이 훨씬 수월해진다.

앱 인벤터에서 알고리즘은 대부분 이벤트 처리로 구현되는데 이벤트 처리 블록 내에 이벤트 처리가 성공적으로 이루어질 수 있도록 필요한 블록들을 순서를 고려하여 조립함으로써 알고리즘이 만들어진다.

HelloPurr 프로젝트의 경우에는 고양이 그림 버튼을 클릭하는 이벤트가 발생했을 때 고양이 울음소리가 나도록 하는 이벤트 처리 한 개만 필요하다.

(Fig. 2) Button Click Event Handler

앱 인벤터에서는 이벤트 처리를 위해 필요한 블록들을 (Fig. 2)와 같이 when - do 블록 내에 조립하는 것으로 이루어진다. HelloPurr 프로젝트가 튜토리얼 중에서 난이도가 가장 낮은 매우 단순한 것이기 때문에 (Fig. 2)와 같은 이벤트 처리가 필요하고 따라서 알고르즘 자체도 사운드 개체에 연결된 소리 파일 하나만 호출하면 되는 것으로 매우 단순하다. 그러나 문제의 복잡

성이 커지면 필요한 이벤트 처리의 수도 늘어나고 알고리즘도 복잡해질 것이다.

알고리즘 간에 또는 알고리즘 내에서도 분해가 가능하고 유사한 패턴이 존재하면 이에 대한 추상화도 가능할 것이다.

5. 결론 및 향후 연구

CT가 컴퓨터과학은 물론 다른 학문 영역에서도 매우 유용하게 활용될 수 있는 사고 과정이라는 것은 이미 학문적으로 널리 인정을 받고 있는 것으로 판단된다. 심지어 어떤 학자는 CT 능력은 이제 읽기, 쓰기, 말하기와 같은 인간의 기본 소양이 되어야 한다고 주장하기도 한다. 문제를 해결하는 사고 과정으로서 이러한 CT의 위상은 장차 미래를 살아갈 학생들이나 이들을 가르치는 교사들에게는 점차 필수적인 역량으로 자리매김할 것이다. 컴퓨터과학자들의 노력으로 만들어진 쉬운 프로그래밍 언어의 사용은 교사 학생 모두가 보다 친근하게 프로그래밍에 접근하는 것을 가능하게 하였다.

이 연구에서 앱 인벤터는 사물(스마트폰)을 직접 조작하면서 학습하기 때문에 주로 컴퓨터 상에서 이루어지는 스크래치 프로그래밍보다 학생들이 추상적 개념을 이해하고 논리적 사고력을 향상시키는데 보다 더 효과적일 수 있다고 주장하였으며, 예비교사들의 실제 수업에서 가장 먼저 다루는 매우 간단한 앱 인벤터 프로젝트를 해결해 나가는 과정에서 CT가 어떻게 구현되는지를 보임으로써 앱 인벤터 프로그래밍 학습이 CT 역량 강화에 도움이 될 수 있음을 규명하였다.

추후에는 CT 역량 평가 방법을 조사하거나 새로운 방법을 개발하여 앱 인벤터 수업 결과 학생들의 CT 역량이 실제로 향상되었는지에 관해서 정량적 평가를 수행하는 연구, 스크래치 프로그래밍과 앱 인벤터 프로그래밍의 학습 효과 비교 연구 등이 이루어질 수 있을 것이다. 또한 CSP(Computer Science Principles)을 가장 잘 반영하기 위한 앱 인벤터 수업 설계에 관한 연구도 이루어질 수 있을 것이다.

참고문헌

[1] Bae, H. J., Lee, E. K., & Lee, Y. J.(2008). A Problem Based Teaching and Learning Model for Scratch Programming Education. *Journal of the Korean Association of Computer Education, 12*(3).

[2] Barr, V., & Stephenson, C.(2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *ACM Inroads, 2*(1).

[3] BBC. BiteSize. Introduction to Computational thinking. http://www.bbc.co.uk/education/guides/zp92mp3/revision.

[4] Byeon, Y. G.(2001). Understanding of the Theory of Teaching and Learning. Hakji Press.

[5] Cho, S. H., Song, J. B., Kim, S. S., & Baek, S. H.(2008). The Effect of a Programming Class Using Scratch. *Journal of The Korean Association of Information Education, 12*(4).

[6] Cho, S. H., Song, J. B., Kim, S. S., & Lee, K. H.(2008). The Effect of CPS-Based Scratch EPL on Problem Solving Ability and Programming Attitude. *Journal of The Korean Association of Information Education, 12*(1).

[7] Choi, H. S.(2014). Developing Lessons and Rubrics to Promote Computational Thinking. *Journal of The Korean Association of Information Education. 18*(1).

[8] Code.org. Why Coding? https://youtu.be/nKIu9yen5nc

[9] Computer Science Teachers Association(CSTA), The International Society for Technology in Education(ISTE).(2011). Computational Thinking - teacher resources(second edition). https://csta.acm.org/Curriculum/sub/CurrFiles/472.11CTTeacherResources_2ed-SP-vF.pdf.

[10] Fletcher, G. H., Lu, J. J.(2009). Education: Human Computing Skills: Rethinking the K-12 Experience. Association for Computing Machiney. *Computations of the ACM, 52*(2).

[11] Google. Computational Thinking for Educators.

https://computationalthinkingcourse.withgoogle.com/course?use_last_location=true.

[12] Google for Education. CT Overview. https://www.google.com/edu/resources/programs/exploring-computational-thinking/index.html#!ct-overview.

[13] Hsu, Y. C., Rice, K., Dawley, L.(2012). Empowering educators with Android App Inventor: An Online Workshop in Mobile App Design. *British Journal of Educational Technology, 43*(1).

[14] Jeong, Y. S.(2015). Proposition for the Software Education Standards of Elementary School. A Public Hearing on the Standard Model for K-12 Software Education.

[15] JTBC(2016.5.17.). Card News.

[16] Kim, S. H., & Han., S. K.(2012). Design-Based Learning for Computational Thinking. *Journal of The Korean Association of Information Education, 16*(3).

[17] Korean Ministry of Education.(2015). Operation Guideline to Software Education.

[18] Lee, E. K.,(2013). Computer Education Curriculum and Instruction: Creative Programming Learning with Scratch for Enhancing computational Thinking. *Journal of The Korean Association of Computer Education, 16*(1).

[19] Mit App Inventor. Computational Thinking through Mobile Computing. https://nsfmobilect.wordpress.com/.

[20] Morelli, R., de Lanerrolle, T., Lake, P., Limardo, N., Tamotsu, E., & Uche, C.(2010). Can Android App Inventor Bring Cumputational Thinking to K-12?, http://www.hfoss.org/uploads/docs/appinventor_manuscript.pdf. Humanitarian Free and Open Source Software Project.

[21] Papert, S(1980). Mindstorms: Children, computers, and powerful ideas. New York: Basic Books.

[22] Song, J. B., Cho, S. H., & Lee, T. W.(2008). The Effect of Learning Scratch Programming on Student's Motivation and Problem Solving Ability. *Journal of The Korean Association of Information Education, 12*(3).

[23] The White House(2016). Computer Science For All. https://www.whitehouse.gov/blog/2016/01/30/computer-science-all.

[24] Turbak, F., Pokress, S. C., & Sherman, M. (2014). Mobile Computational Thinking with App Inventor 2. *Journal of Computing Science in Colleges, 29*(6).

[25] Wing, J. M.(2006). Computational Thinking. *Communications of the ACM, 49*(3).

저자소개

임 병 춘

1979 고려대학교 졸업

1994 고려대학교 졸업(박사)

1985~현재 춘천교육대학교 컴퓨터교육과 교수

관심분야: 컴퓨터교육, 코딩교육

e-mail: blim@cnue.ac.kr