석 사 학 위 논 문

안드로이드 기반의 페이징 기법 시뮬레이터의 설계 및 구현

-FIFO, LRU, Optimal 페이지 교체 알고리즘-

2013년

한성대학교 대학원

컴퓨터공학과

컴퓨터공학전공

구리락차

석 사 학 위 논 문
지도교수 김일민

안드로이드 기반의 페이징 기법 시뮬레이터의 설계 및 구현

-FIFO, LRU, Optimal 페이지 교체 알고리즘-

Design and Implementation for Paging Techniques Simulator in Android-based

2013년 12월 일

한성대학교 대학원

컴퓨터공학과

컴퓨터공학전공

구리락차

석 사 학 위 논 문
지도교수 김일민

안드로이드 기반의 페이징 기법 시뮬레이터의 설계 및 구현

-FIFO, LRU, Optimal 페이지 교체 알고리즘-

Design and Implementation for Paging Techniques Simulator in Android-based

위 논문을 공학 석사학위 논문으로 제출함

2013년 12월 일

한성대학교 대학원
컴퓨터공학과
컴퓨터공학전공
구리락차

구리락차의 공학 석사학위논문을 인준함

2013년 12월 일

심사위원장 조세홍 인

심 사 위 원 장 재 영 인

심 사 위 원 김 일 민 인

국 문 초 록

안드로이드 기반의 페이징 기법 시뮬레이터의 설계 및 구현

-FIFO, LRU, Optimal 페이지 교체 알고리즘-

한성대학교 대학원
컴퓨터공학과
컴퓨터공학전공
구 리 락 차

운영체제 교과목은 운영체제의 구성요소와 동작 방식에 관련된 복잡하고 추상적인 개념들 다루는 교과목이지만, 대부분 교재 위주의 이론 수업으로 진행되어 왔다. 따라서 강의 내용에 대한 이해를 돕고 유발을 위해 교육용 도구의 활용이 지속적으로 시도되고 있다. 본 논문에서는 운영체제의 기억장치 관리 정책에 속하는 페이지 교체 알고리즘들의 올바른 이해를 돕기 위해 페이지 교체 과정을 시각적으로 보여주는 안드로이드 기반의 페이지 교체 시뮬레이터를 설계하고 구현하고자 한다.

학습자는 시뮬레이터를 통해 페이지교체 알고리즘 중 FIFO알고리즘, LFU알고리즘, Optimal 알고리즘에 대한 페이지교체방식과 페이지 부재율을 비교하여 학습하면 텍스트만으로는 이해하기 힘들었던 비가시적이며 추상적인 운영체제 내부의 동작과정이 시각화됨으로써 페이지 교체의 원리를 보다 쉽게 이해할 수 있다. 또한 시뮬레이션의 여러 입력조건들을 학습자가 지접 조절, 선택함으로써 학습자의 능동적인 참여를 유발할 수 있다.

기존의 교육용 시뮬레이션은 학습내용을 전달하는 기능만 강조하여 학습 후 내용에 대한 이해 및 개념의 구조화는 학습자 스스로 분석해야 했기 때문에 학습자 개인의 지적능력에 따라 학습효과가 다르게 나타났다.

본 연구에서는 이러한 문제점을 개선하기 위해 시뮬레이션 학습 직후 학습자

개개인에게 스스로 연습할 수 있는 평가를 제공한다. 시뮬레이터의 비교와 평가 부분은 시뮬레이터를 통해 학습을 하는 동안 학습자가 즉각적인 피드백은 흥미를 유발시킬 수 있게 학습자의 올바른 학습 내용에 대한 이해를 촉진하여 학습효과를 높일 것으로 기대된다.

【주요어】 안드로이드 기반의, 페이징 기법, 시뮬레이터, 운영체제 교과목, 안드로이드

목 차

그 림 목 차

표 목 차

I. 서 론

1.1 연구의 동기

교수학습 방법 중에서 가장 선호하고, 많이 활용되고 있는 방식은 강의식 방법이다. 그러나 교수자가 학습자들에게 강의함에 있어 모든 상황을 효과적으로 전달하는 것은 쉽지 않다. 교수자의 전달 내용을 강의실에서 이해하였음에도 강의실을 나와서 혹은 일정 시간이 지난 후 잊어버리는 경우가 많다. 특히 컴퓨터 교육에 있어서 기본이 되는 교과목인 운영체제(Operation System) 과목은 일반적으로 운영체제 시스템의 구조 및 동작 방법과 관련된 내용을 학습하는 이론 수업으로 진행되고 있다. 본 논문에서는 운영체제 교과목의 내용 중에서 페이징 기법에 대한 이론적 배경을 안드로이드 기반 시뮬레이터를 제작하여 즉각적인 피드백과 흥미유발을 통한 학습자들의 학습 환경을 개선함으로써 학습에 대한 이해도를 향상시키고 한다.

기존의 교육용 시뮬레이션은 학습내용을 전달하는 기능만 강조하여 학습 후 내용에 대한 이해 및 개념의 구조화는 학습효과가 다르게 나타났다.

학습자가 스스로 필요한 지식을 습득하고 정보를 활용하기 위해서는 창의적이고 자율적인 학습이 요구되며, 교사는 학습자 개개인의 학습특성을 고려하여 학습내용 및 학습사태를 개별적이고, 실제 상황에 적용할 수 있도록 제시해주어야 한다. 이러한 개별학습은 컴퓨터를 수업도구로 하여 컴퓨터의 다양한 교수 보조 기능을 활용한다면 교사중심에서 학습자중심으로 변화된 바람직한 수업환경을 조성하고 수업조건을 창출해 낼 수 있다.

1. 2 연구의 목표

컴퓨터를 이용한 학습은 다양한 학습 환경을 제공하지만 일방적으로 행해지므로 창의성을 저해한다는 기존의 우려와는 달리 컴퓨터와 학습자의 상호작용을 가능하게 한다. 왜냐하면 교육용 시뮬레이터를 컴퓨터로 학습내용을 일반적으로 보여 주는 것이 아니라 실제상황과 근접한 학습 환경 속에서 학

습자가 여러 상황 조건들을 직접 조절, 통제하는 과정을 통해 학습이 이루어지기 때문이다. 이러한 컴퓨터와 학습자의 상호작용은 교육용 시뮬레이터를 통해 학습효과를 극대화하고 교과서로 배우는 이론적 지식과 실제 세계의 격차를 좁힐 수 있으며 학습동기 부여, 학습의 전이, 능률성, 비용 절감 등의 이점이 있다. 또한 실제상황에서는 변인들의 통제가 어렵고 너무 복잡하지만 시뮬레이터는 학습에 필요한 부분만을 강조함으로써 학습자가 다른 주변 요소에 의한 혼란을 피할 수 있어 학습이 잘 이루어지게 할 수 있으며 상호작용적이기 때문에 개별화 학습을 위한 효과적인 교수방법이라고 할 수 있다.

운영체제는 컴퓨터 시스템을 구성하는 핵심 요소로써 매우 중요한 학습과목 중의 하나이다. 이러한 운영체제 교과는 컴퓨터의 비가시적인 곳에서 일어나는 현상들을 학습하고 이해해야 하므로 다른 교과에 비해 높은 지적 수준과 인지 특성을 갖은 학습자에게 유리하다고 볼 수 있다. 이에 따라 본 논문은 운영체제 교과의 핵심내용인 가상메모리 관리기법 중 페이지 교체 알고리즘에 대해 가상실험 시스템을 설계하였는데 학습자는 가상 실험을 통해 텍스트 위주의 학습내용을 탈피하고 눈에 보이지 않는 페이지 교체 알고리즘의 동작 과정을 가상의 세계에서 작동시켜 봄으로써 학습에 대한 흥미를 느끼고 페이지 교체 알고리즘의 개념이나 원리를 보다 쉽게 이해할 수 있다.

기존의 알고리즘 학습을 위한 가상실험은 학습내용을 단순히 시각적으로 보여주는데 그쳤기 때문에 학습이 불완전하고 일시적이며, 시뮬레이션학습이 끝난 후 학습자가 직접 개념이나 원리를 재정리해야 하므로 학습자의 지적 수준과 능력에 따라 학습효과가 다르게 나타나는 한계가 있다.

Ⅱ. 이론적 배경

2.1 교육용 소프트웨어

2.1.1 코스웨어

교육용 소프트웨어(Educational Software)는 교육이 지향하는 목적의 달성이나 문제 혹은 과제의 해결을 위한 모든 종류의 소프트웨어를 아울러 이르는 말이다. 코스웨어(courseware)라고도 불린다.

코스웨어는 'course(교육과정)'와 'software(소프트웨어)'의 합성어로 교육 내용과 절차, 방법을 담고 있는 컴퓨터 소프트웨어를 의미한다. 코스에어는 학습방법과 전략에 따른 일련의 학습 절차(instruction)를 담고 있어야 하며 내용은 주로 교육과정에 있는 교과목 내용을 중심으로 하고, 경우에 따라서는 통합교과적인 내용이나 교과에서 벗어난 내용을 포함하기도 한다.

교육용 소프트웨어는 크게 교수 활동을 지원하는 교수용 소프트웨어, 학습 활동을 지원하는 학습용 소프트웨어, 그리고 교육활동 과정이나 제반 문제 해결을 위해서 개발된 교육업무지원 소프트웨어로 구분할 수 있다.

2.1.2 교육용 시뮬레이션의 개념

교육용 시뮬레이션은 컴퓨터를 통하여 실제와 유사한 가상적 상황을 단순화하여 학생들에게 제시함으로써 실제 상황에 관련된 요소나 개념, 원리, 조작 절차, 변화 과정 등을 이해하도록 하는 학습용 소프트웨어의 형태이다. 교육용 시뮬레이션의 학습 상황에서는 학습자가 실제와 유사한 환경에서 학습활동을 수행함으로써 학습자의 능동적인 학습 참여를 유도하고, 실생활에 관련된 개념, 요소, 원리뿐만 아니라 상황진단 및 문제 해결과 의사결정과정 등의 학습의 효과를 증대시킨다. 이러한 시뮬레이션을 통한 수업은 전통적인 교수방법보다 학습자에게 흥미로운 학습 경험을 제시하며 상호작용적인 학습환경을 제공하여 줄 수 있으며 반복학습이 가능하다. 상호작용적인 시뮬레이션 학습자는 문제를 탐구하고 학습자는 인지전략을 응용하며, 다양한 변수의 병인 조작 및 그 영향을 분석하는 과정을 통해 '의미 있는' 학습경험을 할 수

있다.

이러한 교육용 시뮬레이션은 다음과 같은 특징을 가진다.

첫째, 실제 상황과 유사한 상황에서의 여러 가지 적용기술을 습득함으로써 학습의 전이도가 높고 고차원적인 인지 및 사회적/정의적 영역에 긍정적인 효과를 기할 수 있다.

둘째, 복잡하거나 관찰하기 어려운 내용을 명확히 보여줄 수 있을 뿐만 아니라, 실제상황을 가속화시키거나 시간을 지연시킴으로써 특수한 상황에 대한 통찰력과 이해력을 높일 수 있다.

셋째, 학생으로 하여금 학습과정에 능동적으로 참여시킴으로써 다양한 학습활동을 수행할 수 있도록 함으로써 학습동기를 촉진시킨다.

넷째, 위험한 실제 상황을 시뮬레이션 프로그램으로 학습하면 안전하고 편리할 뿐만 아니라 통제가 가능하고 언제나 반복 사용이 가능하며 비용과 시간이 절약되므로 교육의 효율성이 높다. 반면에 교육용 시뮬레이션은 프로그램 제작에게 신중한 설계 계획을 요구하고 학습과제를 설계 하고 제작하는데 많은 시간과 전문성을 요구한다.

2.1.3 교육용 시뮬레이션의 유형

교육용 시뮬레이션은 내용을 가르치기 위한 것과 방법을 가르치기 위한 것으로 분류되며 내용을 가르치기 위한 시뮬레이션은 다시 물리적 시뮬레이션과 과정적 시뮬레이션으로 분류되고, 방법을 가르치기 위한 시뮬레이션과 상황적 시뮬레이션으로 분류된다.

물리적 시뮬레이션은 가장 간단한 현태로써 구체적인 물체나 물리적 대상을 화면에 제공하여 학습자들이 물체 또는 형상에 대해 학습할 기회를 주기 위한 프로그램이다. 예를 들면 빙하의 운동, 대포를 통한 탄도 연구나 자동차 엔진 내부의 연소과정에 관한 시뮬레이션, 비행기 조종시뮬레이션 등이 이에 해당한다. 이러한 물리적 시뮬레이션으로 학생은 대포의 각도가 얼마일 때 물체를 가지고 실험실에서 하는 것보다 적은 노력으로 반복해서 실험을 해 불 수 있다는 이점을 가진다.

과정적 시뮬레이션은 일반적으로 학생들에게 경제 현상이나 수요 공급의 법칙, 인구의 증가와 감소처럼 눈으로 직접 다루지 뭇 하는 과정이나 개념에 대해 알려주기 위해 사용된다. 과정적 시뮬레이션에서 학생들은 시뮬레이션 초기에 다양한 변인을 설정한 다음 컴퓨터가 이 변인들을 상호작용 시켜서 일어나는 과정이 진행되는 것을 보게 된다. 그런 다음에 초기 값을 조정하여 같은 과정이 계속 반복 진행되는데, 변수 값의 변화가 결과에 어떤 영향을 주게 되는가를 관찰함으로써 학습이 이루어지게 된다. 이 시뮬레이션의 특징은 경제 형상이나 인구의 증가와 감소 등 눈으로 직접 볼 수 없는 추상적인 개념이나 과정에 대해 학습할 수 있으며 실제의 과정을 빠르게 또는 느리게 변형시킬 수 있다는 점이다. 예를 들어 요일과 시간대별로 철도의 교통에 대해 분석해야 하는 경우, 실시간으로는 몇 달에 걸쳐 관찰해야 하지만 시뮬레이션에서는 가속된 시간을 사용해 정해진 시간 내에 알아보고자 하는 변화가 어떻게 일어나는지를 눈으로 직접 불 수 있게 해주므로 쉽게 분석할 수 있다.

절차적 시뮬레이션을 어떤 일의 절차 및 순서를 학습하도록 하는 프로그램이다. 컴퓨터가 제시하는 문제 상황에서 학생이 다음 단계에서 취해야 할 행동을 선택하여 반응하면 컴퓨터는 그 반응에 대해 피드백을 제공하고 학습과 컴퓨터의 상호작용이 계속적으로 반복된다. 학생은 축적된 정보를 활용하

여 반전적인 행동을 하게 되며, 횟수가 경과할수록 더 많은 정보를 활용하여 발전적인 행동을 하게 되며, 횟수가 경과할수록 더 많은 정보를 얻게 된다. 예를 들어 의학진단 프로그램의 경우 학습자는 가상의 세계에서 의사가 되어 환자가 찾아와서 여러 증상을 호소했을 때, 환자의 상태를 파악하고자 어떤 의학적 검진을 선택하고 검사결과를 제공하게 되는 것이다. 이러한 절차적 시뮬레이션을 학생이 학습하고 수행해야 할 다양한 방향과 연상효과를 탐구하기 위한 기회를 제공한다.

상황적 시뮬레이션은 규칙과 순서적 절차를 가르치는 절차적 시뮬레이션과는 달리, 다양한 상황에서 일어날 수 있는 인간의 태도와 행동을 다룬다. 상황적 시뮬레이션은 학생들이 접근하는 다양한 방법이 가져오는 효과를 탐구하거나 그 속에서 다양한 역할을 맡아보도록 하여 학생으로 하여금 복잡한 인간행동이나 인간관계에 대한 이해를 높이고 다양한 상황 속에서 문제를 해결할 수 있는 통찰력을 길러준다[2].

2.1.4 교육용 시뮬레이션의 구조와 절차

교육용 시뮬레이션의 구조 및 진행절차는 〈그림 2.1〉과 같다. 시뮬레이션 학습의 진행을 시나리오 제시, 반응요구, 학생의 반응, 피드백과 조절의 순환적인 과정으로 표현하고 있다[1].

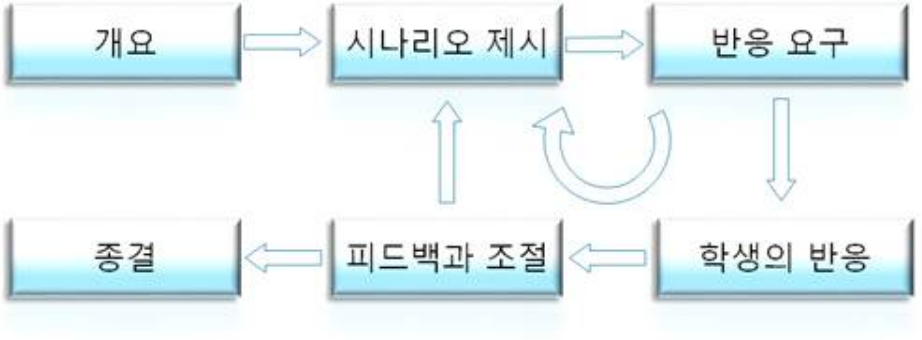

그림 2.1 교육용 시뮬레이션의 구조와 절차

개요 부문에서는 시뮬레이션 형 프로그램 학습에서 일어날 수 있는 여러 가지 사항을 진술하고 학습활동의 최종목표인 학습 목표와 학습자가 조작할 물리적 성질, 학습과정, 지시사항 등을 제시한다. 시뮬레이션에서는 학생이 복잡하고 다양한 활동을 하도록 요구하기 때문에 철저하고 정확한 지시사항이 주어져야 한다. 예를 들면, 학생이 별도의 입력 장치를 사용해야 할 경우 사용방법을 상세하기 설명해 주고 이들을 연습할 수 있는 기회를 주어야 한다.

시뮬레이션의 내용으로 제시되는 체제나 현상에 포함되는 물리적 대상물의 조류와 수에 따라 그 내용의 복잡성 및 정확도가 달라진다. 정확도는 사건이나 상황의 전개가 얼마만큼 정확히 예측될 수 있는가에 달려 있는데, 사물보다 사람이 관련되었을 때 그 내용은 더 복잡해지고 정확도도 낮아지며 매우 어려워진다.

학습자 반응 및 통제는 시뮬레이션에서 학습자가 어떤 행동을 할 수 있는가에 대한 것이다. 시뮬레이션에서 학생의 반응은 키보드, 조이스택, 터치 패널, 마우스, 음성 등 여러 가지 방법으로 입력 장치를 다룰 수 있는 능력 등을 고려해야 한다. 그리고 한 프로그램에서 다양한 형태를 사용하게 함으로써 학습자의 흥미를 증진시키고 학습을 촉진시킬 수 있어야 한다. 학습자들의 반응형태는 여러 선택 대상 중에서 한 가지를 고리거나, 대상물을 조작하거나, 정보의 수집, 정보의 요구 등 다양할 수 있다. 이러한 행동들이 현실 상황에서 요구되는 학생행동과 유사할수록 프로그램의 충실도가 높아진다.

학생의 반응 후에는 그 반응 결과에 대하여 피드백을 받게 된다. 피드백에는 자연적인 피드백과 인위적인 피드백이 있다. 자연적인 피드백은 실제에서 일어난 것과 매우 유사하게 주어지는 것이고, 인위적인 피드백은 실제로 일어난 것과 같은 방식이 아니라, 학습자의 행동에 대해 일어날 수 있는 사태에 대한 정보만을 텍스트와 같이 인위적으로 제시되는 경우를 말한다. 예를 들어, 비행 조정 시뮬레이션에서 안개로 길을 잃었을 경우, 앞에 있는 산에 충돌하는 피드백을 주는 경우는 자연적 피드백이며, “비행기가 산에 충돌했다”라는 메시지를 주는 경우 인위적 피드백이라고 한다. 일반적으로 학생들이 컴퓨터 시뮬레이션 프로그램을 처음 접하는 경우이거나 프로그램의 목적이 학습을 안내하는 것이라면 즉각적이고 정확한 피드백을 주는 것이 바람직하다.

반면에 학생이 모의실험 프로그램에 익숙하거나 내용이 연습이나 평가의 목적으로 사용할 경우에는 자연적 피드백을 주는 것이 효과적이다[3, 4].

2.1.5 교육용 코스웨어에서의 순환학습 모형

교육용 시뮬레이션은 효과적인 학습이론과 컴퓨터의 특징을 고려하여 설계되어야 하는데 순환학습모형은 학생 스스로 구체적인 경험을 통해서 개념을 획득하고, 사고력을 신장할 수 있도록 돕기 위한 탐구 지향적 학습 모형으로 본 논문의 교육용 시뮬레이션 설계에 기본이 되는 것이다.

순환학습모형은 학생들이 능동적으로 참여하여 지식의 구조화 과정에 직접 경험하게 하여 능동적이고 자발적으로 지식을 체계화하게 하는 학습모형이다. 따라서 순환학습모형이란 교육과정 조직의 원리이며 과학의 기본 개념 및 인지 발달을 촉진시키기 위하여 도입한 것으로서 개념화, 구조화, 적용의 순환과정으로 이루어진다. 이 교수학습모형은 3단계가 하나의 고리를 이루어서 이어지고, 이러한 고리가 다시 다음 과정으로 되풀이되기 때문에 순환학습모형이라 불리며 〈그림 2.2〉와 같다.

순환학습의 세 가지 학습 활동 과정을 살펴보면 다음과 같다.

첫 번째는 탐색 단계로 학생들은 새로운 문제 상황에서 스스로의 활동과 그에 따른 반응을 통해 배운다. 이 단계에서 학생들은 기존 개념으로는 해결할 수 없는 인지적 비평 형 상태, 즉 갈등 상황을 경험하며 자율조정을 하거나 상호작용을 통해 문제를 해결하려고 한다. 즉, 새로운 개념의 도임에 대한 필요성이 제기되며 탐구과정이 이 단계에서 수행된다. 이 단계에서 교사는 안내자의 역할만 수행하고 학생 중심의 활동이 이루어 져야 한다.

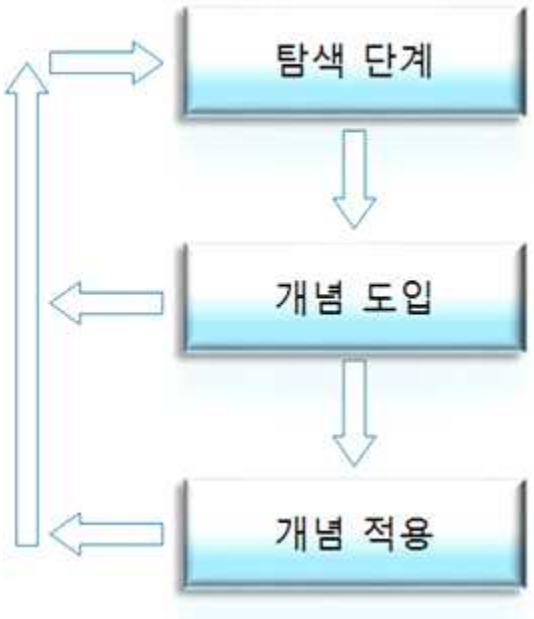

그림 2.2 순환학습모형의 3단계

두 번째는 개념 도입 단계로써 탐색 단계에서의 활동을 통해 가르치고자 하는 개념이나 원리를 교사나 또는 다른 매체에 의해 도입된다. 이 단계에서는 학생들이 학습할 개념은 이미 전 단계에서 경험한 구체적 경험을 내포하고 있어야 효과적이다. 그러나 이전 단계의 경험이 부족한 어떤 학생들은 이 단계만으로 인지적 비평 형을 완전히 해소시킬 수는 없으므로 새로운 상황에 개념을 적용할 기회가 더 많이 필요하다는 점을 교사가 인식하고 있어야 한다.

세 번째는 개념 적용 단계로써 학생들이 새로운 개념 또는 사고형태를 추가적 상황에 적용하는 단계이다. 여기서는 획득한 지식을 응용함으로써 학습자의 인지구조를 안정화시키게 된다. 그러므로 이 단계에서 새로운 개념이 적용되지 않는 상황이 발견되면 또다시 인지적 비평 형을 유발하게 되므로 그 상황에 대한 탐색이 지속적으로 이어지면서 순환학습이 반복되게 된다[3].

2.1.6 순환학습모형을 적용한 코스웨어

본 논문에서는 이러한 순환학습모형의 원리를 가상실험 시스템 설계에 활용하였다. 순환학습모형을 적용한 가상실험시스템의 교수 학습 모형을 살펴보면 〈그림 2.3〉과 같다.

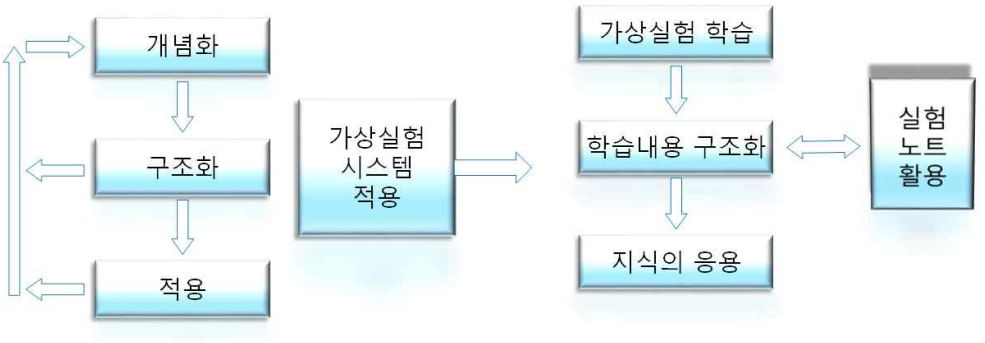

그림 2.3 코스웨어에 적용된 순환학습모형

본 교육용 시뮬레이션에 활용된 순환학습 모형은 다음과 같은 3가지 순서의 학습 사이클을 가져야 한다.

개념화 단계는 교사나 코스웨어로부터 수업을 듣는 것을 의미하며 학생 들은 최소한의 안내로 새로운 자료와 사고를 탐색하면서 환경과의 구체적인 경험을 갖는다. 이 단계에서 교사는 안내자 역할만 하고 실험실에서 실험을 수행하게 하는 것이 좋으며 학습활동은 전적으로 학생들 자신에 의 하여 이루어지고, 생각하며 학습하는 분위기를 조성해야 한다. 순환학습모형에서 개념화 단계는 실험을 통해 학생들 스스로 새로운 개념을 발견할 수 있도록 유도함으로써 학생들의 탐구력을 신장시킬 수 있도록 유도함으로써 학생들의 탐구력을 신장시킬 수 있다.

구조화는 학습자가 학습한 내용을 나름대로 구성하는 단계로 자습이나 복습처럼 남에게 배운 것을 자신의 형태로 재구성하는 것이다. 개념화 단계에서 교사에 의해 제공된 학습내용은 교사에 의 한 학습구조이므로 모든 학습자에게 올바르게 적용될 수는 없다. 이에 학습자는 실험을 통해 얻은 개념을 바탕으로 스스로 경험한 사고 유형을 정의하고 해석하며 학습 방법이나 수준에 개인차가 있으므로 학습내용을 자신의 인지구조에 맞게 재구성할 수 있어야

한다. 이러한 학습의 재구성 방법으로 노트작성이 있는데 이것은 학습내용의 주의집중력을 높여주고 회상에도 도움을 준다. 또한 학습자가 여러 개념들을 결합 시키고 분석하도록 하여 학습한 지식의 수정과 응용을 가능하게 해준다.

그러나 지금까지의 코스웨어들은 이러한 3단계의 코스웨어를 제공하지 못하고 1단계에 머무르는 것이 많으며 2, 3단계에서 다루고 있는 학습자 중심의 교육을 효과적으로 제시하지 못하고 있다. 따라서 일방적인 학습만을 제공하는 1단계 코스웨어는 학습효과를 극대화할 수 없으므로 2, 3단계까지 제공하는 코스웨어의 설계가 필요한 것이다.

2.1.7 알고리즘 시각화의 특징

일반적으로 인간의 지각원리는 시각적 정보가 언어적 정보보다 기억에 오래 남는다. 따라서 그래픽 이미지를 적절히 사용할 경우 수업내용의 의미 전달에 있어서 기대되는 효과는 매우 크다고 할 수 있다.

특히 컴퓨터 관련 분야에서 시각화의 역할은 매우 중요하다. 알고리즘 시각화(Algorithm Visualization)는 텍스트로 기술된 추상적인 형태의 알고리즘을 시각적으로 표현하여 그 수행 과정을 살펴보고 이해하는데 도움을 주는 것으로 알고리즘과 그래픽적 표현 사이의 사상(mapping)이다. 이러한 AV는 단순히 데이터나 프로그램 코드만을 시각화하는 것이 아니라 프로그램 수행과 동시에 내부 자료구조의 변화 및 처리과정을 동적으로 전달하는 것이다. AV는 시각화되는 프로그램의 수행속도나 효율성보다 추상적인 개념을 구체적인 그림으로 표현하여 알고리즘 자체의 내용에 대한 사용자의 이해를 증진시키는 것이 목적이다.

2.2 컴퓨터공학의 운영체제교과

운영체제는 사용자가 작성한 응용 프로그램과 자료를 컴퓨터에 입력할 때, 이 프로그램이 원하는 작동될 수 있도록 지원해 주는 일련의 프로그램들을 의미한다. 즉, 컴퓨터의 핵심을 이루는 소프트웨어의 집단으로서, 컴퓨터를 전공하는 사람은 몰론 컴퓨터를 좀 더 효율적으로 사용하기 원하는 사람들에게는 필수적으로 학습하여야 할 중요한 과목이다.

이러한 관점에서 소프트웨어로 이루어진 컴퓨터의 운영체제는 컴퓨터의 종류나 규모에 관계없이 어떠한 형태로든 항상 존재하여야 하는 것으로, 컴퓨터의 하드웨어를 사용자가 손쉽고 효과적으로 사용할 수 있게 지원해 주는 프로그램의 집단이라고 할 수 있다.

운영체제는 컴퓨터내의 모든 자원을 효율적으로 사용할 수 있도록 하고, 시스템의 성능 향상을 극대화시켜서 전체적인 생산성을 높이는 역할을 담당하는 중요한 요소라고 할 수 있다. 운영체제 과목에서 주로 다루는 내용은 컴퓨터의 주요 자원 중 하나인 CPU의 효율적 관리, 스케줄링, 주 기억 장치에 대한 각 중 관리 기법, 디스크 장치의 관리, 교착 상태, 병행 성 개념과 프로세스간의 관계, 파일 관리, 시스템 자원 보호와 보안 문제 등이 있다.

2.2.1 연구내용과 방법

본 연구에서 가상 실험 학습의 효과를 높이기 위해 실험 노트 기능을 첨가한 교육용 시뮬레이터를 개발하기 위한 것이다. 연구 내용과 방법은 다음과 같다.

첫째, 교육용 시뮬레이터와 코스웨어에 대해 연구한다.

둘째, 기존의 코스웨어를 비교, 분석하고 실제 평가를 제공하는 것이 어떠한 교육적 효과가 있는지 알아본다.

셋째, '페이지 교체 알고리즘' 학습을 위한 시뮬레이션 형 코스웨어를 구현하고. 시뮬레이션 학습 후 실험과정 및 결과를 체계적으로 정리할 수 있는 평가를 자동 생성한다.

넷째, 실험 실습을 통해 가상 실험 학습에서 평가가 어떻게 작성되는지 분석한다.

정보통신기술의 교육적 활용으로 인해 교육에서도 교사의 역할은 이제 더 이상 '가르치는 자' 로서만 국한되지 않는다. 대신 학생들의 학습을 도와주고, 학습에 직접 참여함으로써 '같이 배우는 자' 로서의 역할을 경험하는 시대가 되었다.

운영체제 과목은 분산 처리 과목을 비롯한 고급 개념에 대한 대응력 배양과 분산 시스템, 분산 운영 체제에 대한 적응력을 길러주며, 운영체제의 기본 원리에 대한 시스템 프로그래밍 구사 능력의 향상으로 컴퓨터 전문가로서 성장할 수 있는 밑거름이 될 수 있는 과목이다.

2.2.2 페이지 교체 알고리즘

페이지 폴트가 발생하면 운영체제는 새로 진입할 페이지를 위한 공간을 만들기 위해 이미 존재하고 있는 페이지 중에 하나를 내 보내야(메모리에서 제거해야)한다. 만일 내 보낼 페이지가 변경되어 있다면 그 페이지의 내용은 디스크로 보내져 기록되어야 한다. 그래야만 디스크에 있는 쉐도우(shadow) 페이지가 가장 최근의 데이터를 유지할 수 있다. 반면, 만일 페이지가 변경되지 않았다면(예를 들어 프로그램의 텍스트 부분을 가지고 있었다면) 디스크에 기록 할 필요는 없다. 새로 읽혀진 페이지는 쫓겨난 페이지가 차지하고 있던 공간을 차지한다.

내보낼 페이지를 임의적(random)으로 선택할 수 있다. 하지만 임의적으로 선택하는 운영체제 교과목의 학습 내용 중에서 FIFO(First In First Out), LRU(Least Recently Used), LFU(Lest Frequently Used), MFU(Most Frequently Used), OPT(OPTimal)와 같은 페이징 기법을 각각 하나의 페이지 단위로 구현되는 모습을 보여줌으로써 학습자가 각 페이징 기법의 구현 방법을 쉽게 익힐 수 있도록 설계하였으며, 페이징 기법들 간의 비교를 통해 성능을 측정할 수 있도록 하였다. 특히, 학습자의 학습 이해 정도를 측정하기 위한 평가 시스템을 추가하여 스스로 자신의 학습 정도를 평가할 수도 있다.

가장 간단한 페이지 교체 알고리즘은 FIFO(First-In-Out) 교체 알고리즘이다. 선입선출 교체 알고리즘은 각 페이지에 메모리 안으로 들어온 시간을 이용하는데, 가장 오래된 페이지부터 우선 교체시킨다. 페이지 부재가 발생하면, 즉 제거 해여 할 페이지를 선택하면 보조기억장치로 이동 교체시키고 페이지

테이블의 타당/비트당 비트를 변경한다. 새로운 페이지에 대한 페이지 테이블 항목을 변경한 후, FIFO 큐의 마지막 위치에 삽입한다.

메모리에 있는 모든 페이지는 선입선출 큐(FIFO queue)에 의해 관리된다. 따라서 큐의 헤드부분에 있는 페이지를 먼저 대치시킬 수도 있다. 페이지가 메모리 속으로 들어올 때 큐의 끝에 페이지를 삽입한다. 이때 큐의 크기는 사용 가능한 메모리 프레임의 수에 해당된다.

예를 들어, 세 개의 프레임을 사용할 수 있다고 가정하고 위 참조 문자열을 실행시켜 보면 15개의 페이지 부재가 발생한다.

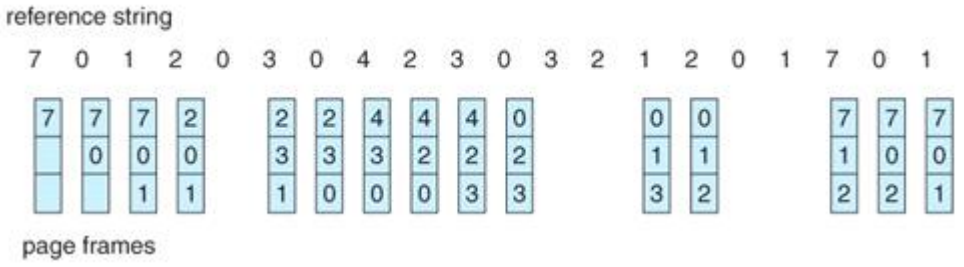

표.1 FIFO 알고리즘의 실행 결과 /출처: http://www.cs.uic.edu

처음에는 세 개의 프레임이 비어 있다. 그리고 처음 세 개의 참조는 페이지 부재를 일으켜서 빈 프레임 속으로 들어간다. 그 다음의 참조 문자열 2는 처음에 들어왔던 페이지 7을 교체한다. 그 다음 참조 문자열 0은 이미 메모리에 있기 때문에 페이지 부재가 발생하지 않는다.

참조 문자열 3에 대한 참조는 페이지 0에 의해 대치된다. 왜냐면 메모리에 들어있는 세 개의 페이지(0, 1, 2) 중에 페이지 0이 가장 빨리 들어왔기 때문이다. 이런 교체는 다음 참조인 0이 페이지 부재가 된다는 것을 의미하며, 가장 빨리 들어온 페이지 1과 교체된다. 이러한 과정은 그림과 같이 계속 진행된다.

선입선출 교체 알고리즘은 이해가 쉽고 프로그램 작성도 쉽다는 장점이 있으나 성능이 항상 좋은 것은 아니다. 선입선출 교체 알고리즘에서 발생할 수 있는 문제점을 알아보기 위해 다음과 같은 참조 문자열을 생각해 보자.

참조 문자열: 0, 1, 2, 3, 0, 1, 4, 0, 1, 3, 4

프레임이 많으면 페이지 부재 횟수가 줄어드는 현상에 반대되는 현상이 나타난다. 즉, 프레임이 3개일 때보다 프레임이 4개일 때 페이지 부재가 많이 발생하는데, 이러한 현상을 벨레디의 변이(Belady's Anomaly)라고 한다[6].

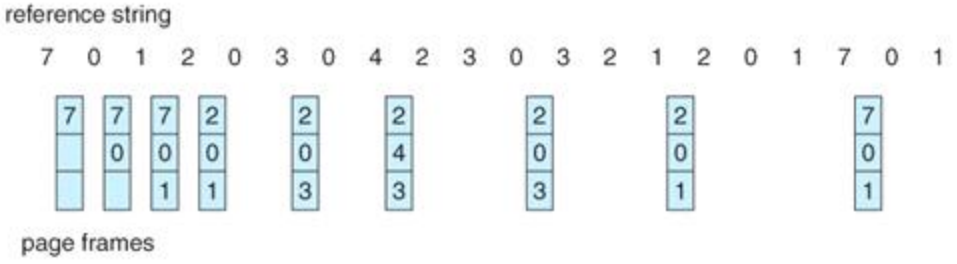

표.2 Optimal 알고리즘의 실행 결과 /출처: http://www.cs.uic.edu

예를 들어, 앞의 참조 문자열에서 최적 페이지 교체 알고리즘은 그림과 같이 9번의 페이지 부재를 발생시킨다. 처음 3번의 참조는 부재를 일으키고 3개의 비어 있는 프레임을 채우는 과정은 동일하다. 최적의 페이지 교체 알고리즘은 가장 큰 레이블을 갖는 페이지를 교체한다. 만일 8백만 명령어 뒤에 참조되는 페이지와 6백만 명령어 뒤에 참조되는 페이지가 있다면, 앞에 있는 페이지를 교체하게 되며, 결국 교체에 따른 페이지 폴트의 발생을 가능한 많이 뒤로 미룰 수 있게 된다. 컴퓨터도 사람과 마찬가지로 반갑지 않은 이벤트의 발생은 가능한 뒤로 미루려고 노력한다.

앞으로 가장 오랫동안 사용되지 않을 페이지를 교체하므로, 최소의 페이지 부재율을 보이는 기법이다. 그러나 이 기법은 페이지 호출 순서에 대한 모든 상황을 미리 파악하고 있어야 하므로 비현실적이다. 이 알고리즘의 유일한 단점은 구현이 불가능하다는 것이다. 페이지 폴트가 발생했을 때 운영체제는 각 페이지들이 미래의 어느 시점에 참조될 지 알 수 없다. 이 알고리즘은 프로그램을 시뮬레이터에서 수행하고, 모든 페이지 참조 정보를 모으고, 두 번째 수행에서는 수집한 정보를 기반으로 최적의 페이지 교체 알고리즘을 구현하는 것이다.

이러한 방법으로 최적의 알고리즘과 실제 구현 가능한 알고리즘과 성능 비교가 가능하다. 만일 운영체제가 현재 사용하는 알고리즘이 최적의 알고리즘

에 비해 1% 성능이 나쁘다면 더 좋은 알고리즘에 대한 노력은 최대 1%의 성능 향상을 가져 올 수 있을 뿐이다.

LRU(Least Recently Used)는 가장 많이 사용되는 교체 기법으로써 페이지 참조의 시간적 지역성을 고려하여 FIFO 기법의 이상 현상을 해결하고 OPT 기법의 예측에 대한 문제점을 개선하기 위하여 제안된 것이다. 이 기법은 어떤 페이지가 참조되면 가까운 장래에 그 페이지가 다시 참조된다는 시간적 지역성을 고려하여 가장 최근에 참조되지 않은 페이지를 교체의 대상으로 선택하는 것이다. 적재된 페이지들 중에서 가장 오래 전에 참조된 페이지를 교체의 대상으로 선택하는 것이다. 〈표.3〉에 LRU 알고리즘 실행 결과를 표시된다.

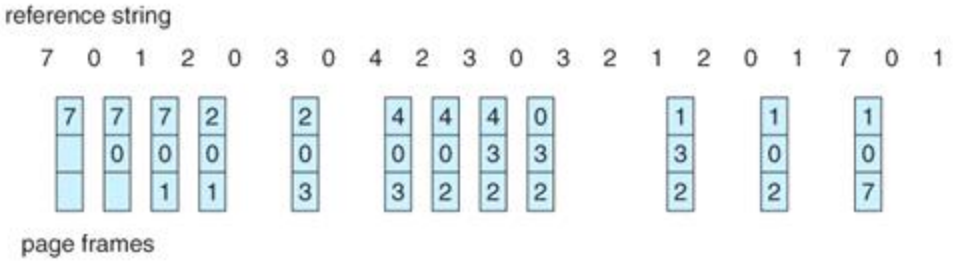

표.3 LRU 알고리즘의 실행 결과 /출처: http://www.cs.uic.edu/

페이지마다 타임스탬프용 카운터를 두어 현 시점에서 가장 오랫동안 사용되지 않은 페이지를 교체하는 기법이며, 가장 널리 사용됩니다. 이 기법은 오랫동안 참조되지 않은 페이지는 가까운 장래에도 사용되지 않을 것이라는 가정에 근거하지만, 참조 시간을 기록해야 하므로 오버헤드가 발생하고 구현하기가 매우 복잡하다는 단점이 있습니다.

LRU 알고리즘은 구현 비용이 많이 든다. LRU를 위해서는 메모리 내에 있는 모든 페이지들이 리스트로 연결되어 있어야 하며, 가장 최근에 참조된 페이지는 리스트의 앞에 그리고 가장 과거에 참조된 페이지는 리스트의 뒤에 위치해야 한다. 이때 어려운 것은 모든 메모리 참조마다 리스트를 갱신해야 한다는 것이다. 리스트에서 페이지를 찾고, 삭제하고, 리스트의 가장 앞으로 이동하는 작업은 시간이 많이 걸리는 작업이다. 심지어 하드웨어 적으로 동작해도 부하기 크다(이런 하드웨어를 구현할 수 있다고 가정한다면).

LRU를 리스트가 아닌 다른 특별한 하드웨어의 지원으로 구현할 수도 있다. 우선 아주 간단한 방법부터 살펴보자. 이 방법은 C라는 64비트 카운터를 필요로 한다. 이 카운터는 명령어가 실행될 때마다 1씩 증가한다. 한편 각 페이지 테이블 엔트리는 카운터 값을 저장할 수 있는 공간을 가지고 있다. 메모리가 참조될 때마다 참조된 메모리를 담고 있는 페이지를 가리키는 페이지 테이블 엔트리에 C의 값이 저장된다. 페이지 폴트가 발생하면 운영체제는 모든 페이지 테이블 엔트리의 카운터 값을 조사하여, 가장 적은 값을 갖는 페이지를 교체한다. 이 페이지가 LRU 페이지이다.

Ⅲ. 시뮬레이터의 설계

3.1 시뮬레이터의 설계 및 구현

실행 환경: 안드로이드 SDK 2.3
사용 언어: 프로세싱 PDE 2.0 Beta 6
구현 내용: 과목 - 운영체제, 내용 - 페이징 교체 기법

3.2 프로세싱 프로그래밍 언어

3.2.1 프로세싱 프로그래밍 언어

시뮬레이터를 설계할 때 **프로세싱 언어**를 사용하며 쉽고 구현할 시간을 줄이다. 2001년에 MIT(Massachusetts Institute of Technology)에서 Ben Fry와 Casey Reas으라는 두 연구원들이 매디어 예술을 쉽게 만들 수 있는 프로세싱으라는 프로그래밍 언어를 만들었다. 프로세싱 언어는 DbN(Design by Numbers, MIT)에 영향을 받았으며 프로그래밍 경험이 적은 디자이너나 예술가들에게 프로그래밍을 가르치기 위한 교육용 언어다. DbN은 쉽게 배울 수 있으나 기능이 제한되어 있었으며, 이를 보완 확장한 프로세싱은 2002년 8월 알파버전이 만들어진 후 2008년 processing 1.0이 공개되었다. 디자이너, 미디어 아티스트들이 프로토타입을 만들거나 작품을 제작하는 용도로 가장 많이 사용하고 있다. 프로세싱 갤러리 http://processing.org/exhibition에서 작품을 감상할 수 있다.

프로세싱 프로그래밍 언어는 초보자나 디자이너, 미디어 아티스트들이 쉽게 사용할 수 있도록 개발되었다. 프로세싱에 대한 자세한 정보는 http://processing.org 사이트에 있다. 그래픽이나 이미지, 비디오, 사운드를 쉽게 다룰 수 있는 명령어들은 내장하며, 라이브러리를 이용해 3D나 PDF, Video 출력 등을 쉬게 다룰 확장할 수 있다. 자바 기반이기 때문에 Windows, Linux, Mac OS 등의 어느 플랫폼에서도 자바가 설치되어 있으면 동일한 코드의 실행이 가능하다[10].

http://processing.org/download 사이트에서 프로세싱의 고식적인 다운 받

을 수 있다. 프로세싱 개발 환경 PDE(Processing Development Environment)라고 한다. 여기서 스케치, 컴파일, 실행을 모두 할 수 있다. 프로세싱은 간단한 편집기가 있다. 여기서 스케치를 만들면서 프로그램 코드를 쓰고 실행된다. 생각은 스케치를 하면서 정리되고 스케치하는 과정은 놀이처럼 재미있으면서도 간편하다. 스케치의 기본적인 목표는 짧은 시간 안에 많은 아이디어들을 탐구 하는 것이다. 우리 저자들의 경우 조 이에 스케치를 한 뒤에 그 결과를 코드로 옮긴다. 애니메이션과 인털액션에 대한 아이디어는 스토리보드에 지문을 더하는 방식으로 스케치한다. 소프트웨어 스케치를 어느 정도 한 뒤에는 가장 좋은 아이디어들을 선택하고 결합하여 프로토타입을 만든다. 이는 종이와 스크린을 왔다 갔다 하며 만들고, 시험하고, 개선하는, 일종의 순환적 과정이다.

마치 여러 소프트웨어가 담겨있는 공구함처럼, 프로세싱을 구성하는 툴은 다양한 조합을 이루며 쓰일 수 있다. 그러므로 프로세싱은 간단한 작업뿐만 아니라 복잡하고 심층적인 연구에도 사용될 수 있다. 프로세싱 프로그램은 한 줄 정도로 짧을 수도 있고 수천 줄 정도로 길어질 수도 있기 때문에 성정과 변화의 여지가 있다. 라이브러리가 100개 이상인 덕분에 프로세싱은 사운드, 컴퓨터 비전, 디지털 공정의 영역까지 확장된다.

3.2.2 프로세싱과 자바

인간의 언어와 같이 프로그래밍 언어도 관련된 어족이 있다. 프로세싱은 자바 프로그래밍 언어의 방언이다. 언어 구문은 거의 같지만 프로세싱에는 그래픽 및 인터랙션과 관련하여 사용자 정의 기능들이 추가됐다. 프로세싱의 그래픽 요소들은 PostScript(PDF의 기반) 및 OpenGL(3D 그래픽 설계서)과 관련이 있다. 이처럼 여러 언어의 특징을 공유하고 있는 덕에, 프로세싱을 배운다는 것은 다른 언어들로 프로그래밍하는 과정에 입문하는 것과 같으며 또한 여러 소프트웨어 툴을 혼용하는 일이기도 하다[7].

프로세싱의 개발환경을 자바 언어로 썼다. 프로세싱 프로그램을 먼저 자바 언어로 번역해서 자바 프로그램처럼 실행된다. 자바보다는 프로세싱이 활씬 접근하기 쉬운 구조로 되어있고, 그렇기 때문에 인더랙티브 그래픽 프로젝트를 만드는 데 적합한 툴이기도 하다. 다음은 자바 프로그래밍에는 없고 프로

세싱에서만 제공하는 것들이다.

- 도형을 만들 수 있는 간단한 함수
- 텍스트, 이미지, 비디오를 불러올 수 있는 간단한 함수
- 3D 환경을 만드는 함수
- 마우스와 키보드 인터랙션을 가능하게 하는 함수
- 코드를 간단히 생성할 수 있는 개발 환경
- 아티스트, 디자이너, 프로그래머로 구성된 온라인 커뮤니티 활성화!

그런데 자바에서는

- 변수의 선언, 초기화, 사용을 같은 방법으로 한다.
- 배열의 선언, 초기화, 사용을 같은 방법으로 한다.
- 조건문과 반복문 또 같은 방법으로 한다.
- 함수의 선언과 호출을 같은 방법으로 한다.
- 클래스의 생성도 같은 방법으로 한다.
- 객체 생성도 같은 방법으로 한다.

프로세싱에서 재생 버튼을 누르면 코드가 실행되는데, 이때 첫 번째 과정이 프로세싱 코드를 자바로 번역하는 것이다. 이와 함께 코드를 애플릿으로 만들 때 컴파일 과정을 거쳐서 스케치 폴더 안에 '스케이치이름.pde' 파일과 함께 '스케치이름.java' 파일이 생성된다. 이 자바 파일이 바로 자바 언어로 번역된 파일이고 자바로 번역된 파일을 열어 보면 아주 작은 변화가 있다.

프로세싱에서 화면에 그림을 그리는 함수의 핵심 라이브러리는 시각적인 피드백과 단서를 주어 코드가 무엇을 하고 있는지 알 수 있게 해준다. 그리고 이 프로그램 언어는 다른(특히 JAVA(자바))들의 이론, 구조, 문법 등을 따르기 때문에 여러분이 프로세싱에서 배우는 모든 것은 실제 프로그래밍이다. 다시 말해서 특정 소프트웨어에서만 해석 가능한 스크립트 언어가 아니기 때문에 프로세싱은 다른 모든 프로그램 언어들이 가지고 있는 기초와 핵심 부분을 가지고 있다.

프로세싱 개발 환경은 컴퓨터 코드를 작성하기 쉽도록 단순화된 환경이다.

그리고 이것은 간단한 텍스트 편집 소프트웨어(메모장 같은)가 미디어 플레이어와 결합한 형태라고 생각하면 된다. 각 스케치(프로세싱 프로그램은 '스케치(sketch)'로 불린다)는 파일 이름, 여러분이 코드를 타이핑할 수 있는 공간, 그리고 저장, 열기, 스케치를 실행할 버튼을 가지고 있다. 스케치 메인 화면이 〈그림 3.1〉과 같다.

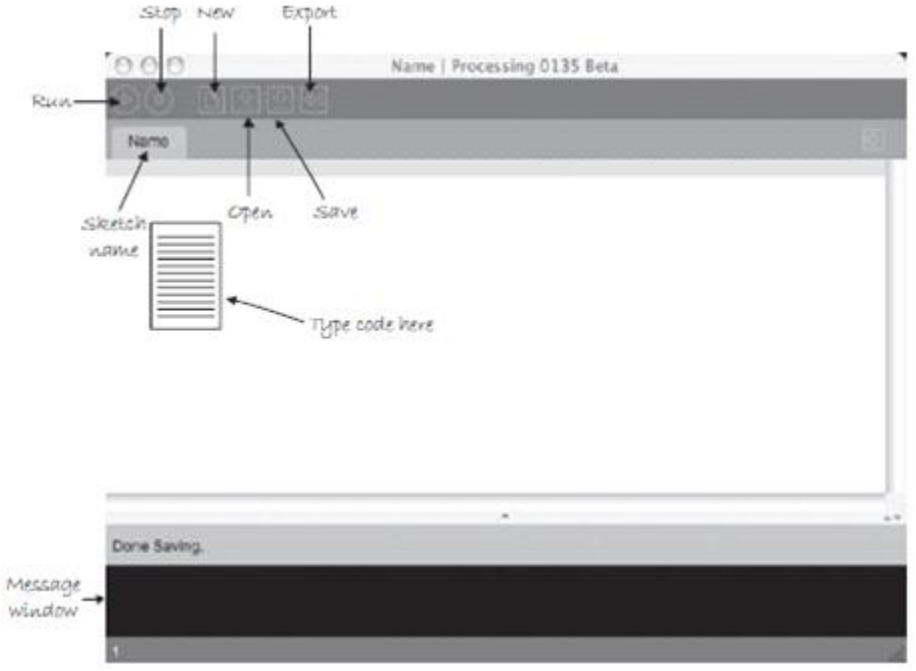

그림 3.1 프로세싱 스케치 /출처: www.processing.org/

프로세싱 프로그램은 비공식적으로 스케치(sketch)라고 부르는데 이는 빠른 시각적인 프로토타입을 만들 수 있다는 의미이다. 스케치를 저장한 폴더를 '스케치북'이라고 부른다. 기술적으로 말하자면, 프로세싱에서 스케치를 웹 프로그램(웹 브라우저에서 실행되는 작은 프로그램)으로 출력할 수 있게 해주고 독립된 프로그램으로 독자적으로 실행되는 형태(예를 들어 다운로드 가능한 프로그램)로 만들어 주기도 한다. 프로세싱의 예제가 실행되는 것을 확인했다면 자신만의 스케치를 만들 준비가 된 것이다. 'New' 버튼을 클릭하면 오늘 날짜로 파일 이름이 붙여진 새로운 빈 스케치가 만들어질 것이다. 'Save as' 버튼을 눌러 여러분만의 스케치 이름으로 바꾸어 저장하지(주의: 스케치 이름으로는 띄어쓰기나 하이픈, 숫자로 시작하는 이름을 사용할 수 없다).

여러분이 초기 프로세싱을 실행시켰을 때 프로세싱에서 생성된 모든 스케치들은 윈도우의 겨우 '내 문서' 폴더에, OS X일 경우 '문서'에 저장되게 되어 있다. 이것은 기분설정이며, 여러분이 원한다면 하드 드라이브의 다른 위치에 저장할 수 있다.

각 프로세싱 스케치는 폴더(여러분의 스케치와 동일한 이름)로 이루어져 있으며, 파일명에 확장 'pde'가 붙어 있다. 만일 여러분이 프로세싱 스케치에 MyFirstProgram이라는 이름을 붙여 이었다면, 폴더는 MyFirstProgram으로 이름 지어질 것이고, 그 안에는 MyFirstProgram.pde 파일이 있을 것이다. 그 'pde' 파일은 소스코드를 포함한 문자 파일이다. 또한 몇몇 스케치에는 'data'라는 폴더가 있는데 이 안에는 프로그램에서 사용된 이미지 파일, 사운드 클립과 같은 미디어 요소가 저장된다.

3.2.3 프로세싱 코딩

우리가 작성할 수 있는 세 가지 형태의 문장이 있다.

- 함수 호출
- 연산자 대입
- 제어 구조

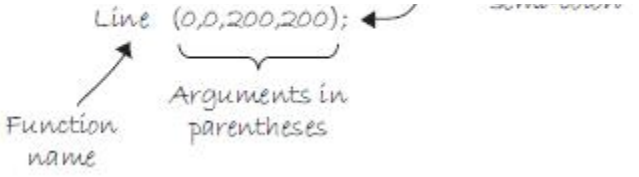

그림 3.2 프로세싱 문법 /출처: www.processing.org/

모든 줄의 코드는 함수를 호출하는 것이다. 함수는 이름을 가지고 있고 괄호로 닫힌 한 세트의 인자를 수반한다. 프로세싱은 함수들의 배열을 하나씩 실행 할 것이고 화면에 결과물을 나타낼 것이다.

3.2.4 Setup()과 draw() 함수

Setup()과 draw() 함수는 프로세싱의 중심 함수이다. 애니메이션과 상호 작용을 제공하기 위해 프로그램을 setup()과 draw() 함수를 사용하여 대화 형 모드로 작

성한다. setup()은 프로그램 시작할 때 한 번만 실행되는데 draw()는 프로그램이 종료될 때까지 반복된다. size()으나 (화면 크기) 초기 값 등을 여기 선언하는 것이 좋다. draw()는 기본적으로 초당 60프레임을 실행하며 함수는 괄호 내의 콘솔 창에 나타내기 위해 사용된 내장함수이다.

그림 3.3 프로세싱 개발 환경/PDE/ (예제: 공이 오른 쪽 계속 돌아간다).

출처: www.processing.org

3.2.5 프로세싱과 안드로이드

세계 각국 이동통신 관련 회사 연합체인 OHA(Open Handset Alliance)가 개발하여 2007년 11월에 공개하였는데, 실질적으로 세계적 검색엔진 업체인 구글(Google)이 주도하였으므로 '구글 안드로이드'라고도 한다.

그림 3.4 프로세싱과 스마트폰

안드로이드는 리녹스2.6 커널을 기반으로 강력한 운영체제와 포괄적 라이브러리 세트, 풍부한 멀티미디어 사용자 인터페이스, 폰 애플리케이션 등을 제공한다. 휴대폰뿐 아니라 다양한 정보 가전 기기에 적용할 수 있는 연동성도 갖추고 있다.

안드로이드가 기존의 휴대폰 운영체제인 Windows Mobile이나 Symbian과 차별화되는 것은 완전 개방형 플랫폼이라는 점이다. 소스 모드를 모두 공개함으로써 누구라도 이를 이용하여 소프트웨어와 기기를 만들어 판매할 수 있도록 하였다. 개발자들은 이를 확장, 대체 또는 재사용하여 사용자들에게 풍부하고 통합된 모바일 서비스를 제공할 수 있게 된 것이다.

안드로이드를 탑재한 휴대폰 단말기를 안드로이드폰이라고하며, 이 플랫폼에서 응용할 수 있는 애플리케이션을 거래하는 온라인 공간을 플레이 스토르라고 한다.

개발 언어는 자바형식의 자체적인 API를 지원하고 자바 API의 일부분을 지원한다. C와 C++로 개발 된 외부 라이브러리를 제약적으로 사용이 가능하다.

안드로이드는 email 클라이언트, SMS 프로그램, 달력, 지도, 웹 브라우저 등을 포함하는 핵심 프로그램들을 기본적으로 포함하고 있다. 모든 프로그램은 자바 언어로 만들어졌다.

프로세싱 프로그램을 자바 소스 코드로 옮겨서 자바 프로그램을 만들 있는 것처럼 안드로이드 애플리케션을 쉽게 만들 수 있다. 안드로이드 SDK를 먼저 설치해야 한다. developer.android.com/sdk/index.html 사이트에서 무료 다운로드 가능하다. Android 2.3.3 (API 10), SDK Platform, Google APIs by Google Inc, 곡 설치되어 있어야 한다.

PDE(Processing Development Environment)는 Java, JavaScript, Android 모드를 가지고 있고 이 중에서 Android mode를 고르면 된다. 다음과 같은 방법으로 시행할 수 있다.

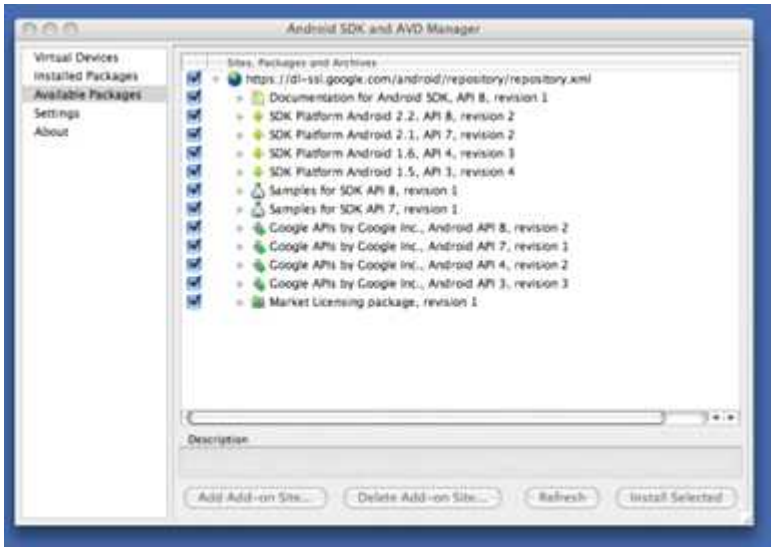

그림 3.5 Android SDK and AVD Manager

- Run in Emulator - 안드로이드 에뮬레이터 모드로 실행.
- Run on Device - 장치에서(스마트폰) 실행.
- Export Android Project - 안드로이드 프로젝트 폴더를 만들기.

3.3 시뮬레이터의 설계

1. 교체정책: 교체정책 이론 습득.
2. 본 코스웨어의 교체정책 화면에 교체정책 개념, 교체정책 종류, 페이지부재율 계산식들이 구성되어 있고 각각 터치를 하면서 학습할 수 있게 구현했다.

그림 3.6 시뮬레이터의 메인 화면. /터치하면 교체정책 화면으로 들어간다. /

교체정책 화면에서 교체 정책 개념, 교체정책 종류, 페이지 부재율 계산식에 대한 각각 설명을 들어 가 볼 수 있게 만들었다.

그림 3.7 교체 정책 개념, 교체 정책 종류, 페이지 부재율 계산식

3.3.1 페이징 기법(FIFO, LRU, OPT)

페이징 기법 시뮬레이션 구동, 교체할 때, hit가 되거나 교체될 때 색갈이 바뀐다. 학습자가 원하는 알고리즘별로 페이지 교체가 일어나는 선서의 차이와 페이지 부재율의 차이를 확인하도록 하였다.

그림 3.8 FIFO 알고리즘 선택

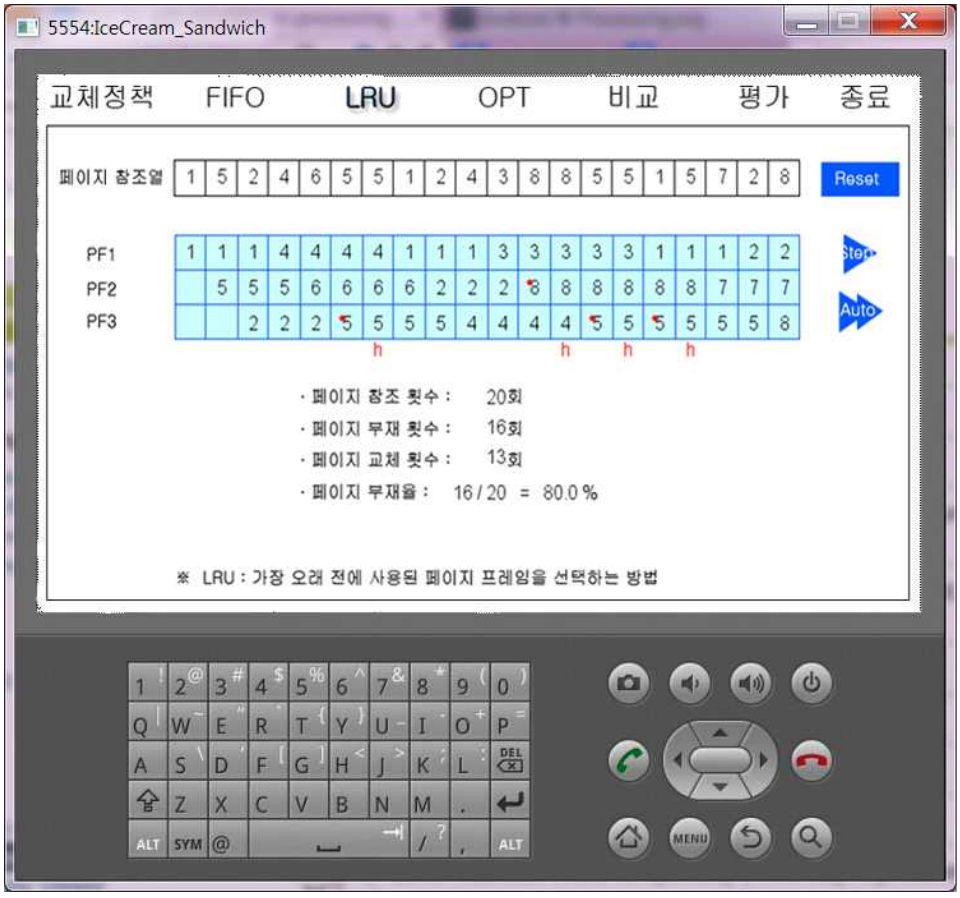

그림 3.9 LRU 알고리즘 선택.

그림 3.10 Optimal 알고리즘 선택.

화면 위에 있는 교체정책, FIFO, LRU, OPT(Optimal), 비교, 평가, 종료 선택들을 터치 하면서 알고리즘 들을 실행하고, 비교하고 스스로 평가할 수 있게 만들었다. 알고리즘 선택마다 다음과 실행하는 버튼들 있다.

Reset 랜덤을 터치하면 생성된다.

Step 버튼을 터치하면 한 단계 식 실행된다.

Auto 버튼을 터치하면 한 번에 다 실행된다.

4. 평가: 페이징 기법에 대한 이해도를 스스로 평가.
5. 종료: 프로그램을 종료하기.

3.3.2 비교: 페이지 기법들의 성능 비교.

FIFO, LRU, OPT 알고리즘을 한 화면에서 애니메이션 해 봄으로써, 시각적으로 알고리즘의 성능을 비교 할 수 있도록 하였다. 각 각의 알고리즘이 변화되는 모습을 확인할 수 있으며, 페이지 교체가 일어나야 할 때 알고리즘별로 차이가 남을 확인 할 수 있다. 페이지 부재율을 표시하여 알고리즘별로 성능을 숫자적으로 확인할 수 있도록 구현하였다. 이 선택에서는 FIFO, LRU, Optimal 세 알고리즘을 같이 한 단계씩, 한꺼번에 볼 수 있다.

그림 3.11 페이지 교체 기법, 비교 선택

3.3.3 스스로 평가

학습평가를 두어 학습자가 스스로 학습결과를 평가할 수 있는 기회를 제공함으로써 학습효과를 높일 수 있도록 하였으며, 페이징 기법, 교체정책 각 각에 대해 구성하였다. 서브메뉴로 1~5까지의 버튼을 두어 원하는 문제로 자유롭게 이동할 수 있도록 하였다. 문제는 학습목표에 관련된 내용으로 하였으며, 학습내용을 충실하게 학습자는 충분히 풀 수 있는 문제로 구현하였다.

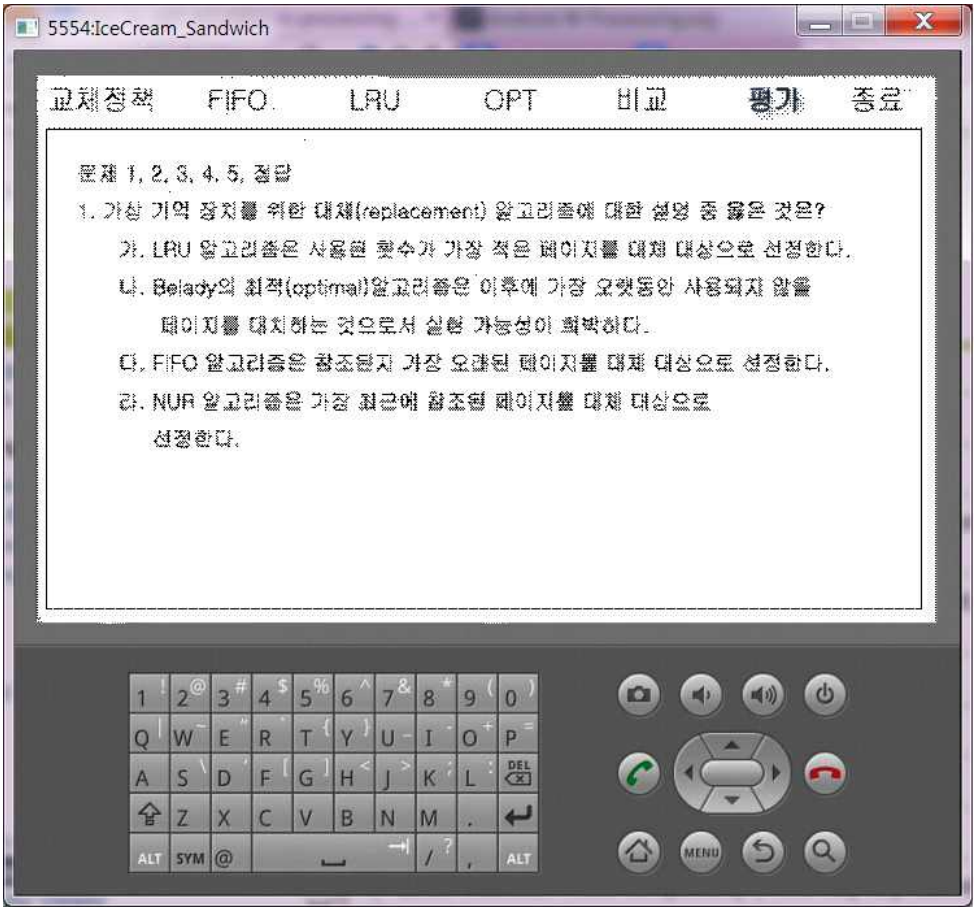

그림 3.12 스스로 평가

3.4 시뮬레이터 구현

3.4.1 주요 구현 함수

- setup(): 초기 값 설정.

프로그램의 함수들을(pageFrame 클라스의load_fifo(), load_lru(), load_opt(), load_compare() 함수들) 선언하고 초기 값들을 설정하는 부분이다. 텍스트와 이미지 파일을 배열로 여기에서 선언했다.

- draw(): 화면 draw.

draw() 함수는 프로그램이 종료될 때가지 계속 반복하기 때문에 이 안에서 페이지 참조 횟수, 페이지 부재 횟수, 페이지 교체 횟수 등 페이지 프레임의 모든 출력하는 부분들이 있다.

- mouseClicked(): 상단 메뉴 및 각 메뉴 별 버튼 제어.
  터치하면 어떤 작업을 하는지 제어하는 부분이다.
- createMessageBox(): 종료 메시지 박스.
  프로그램을 종료할 때 나오는 메시지 박스다. Yes 혹은 No 선택이 있다.
- class pageFrame: FIFO, LRU, OPT 알고리즘, 비교 부분.
  pageFrame 클래스에서 각 알고리즘의 데이터를 생성하는 load_fifo(), load_lru(), load_opt(), load_compare() 함수들이 있다.

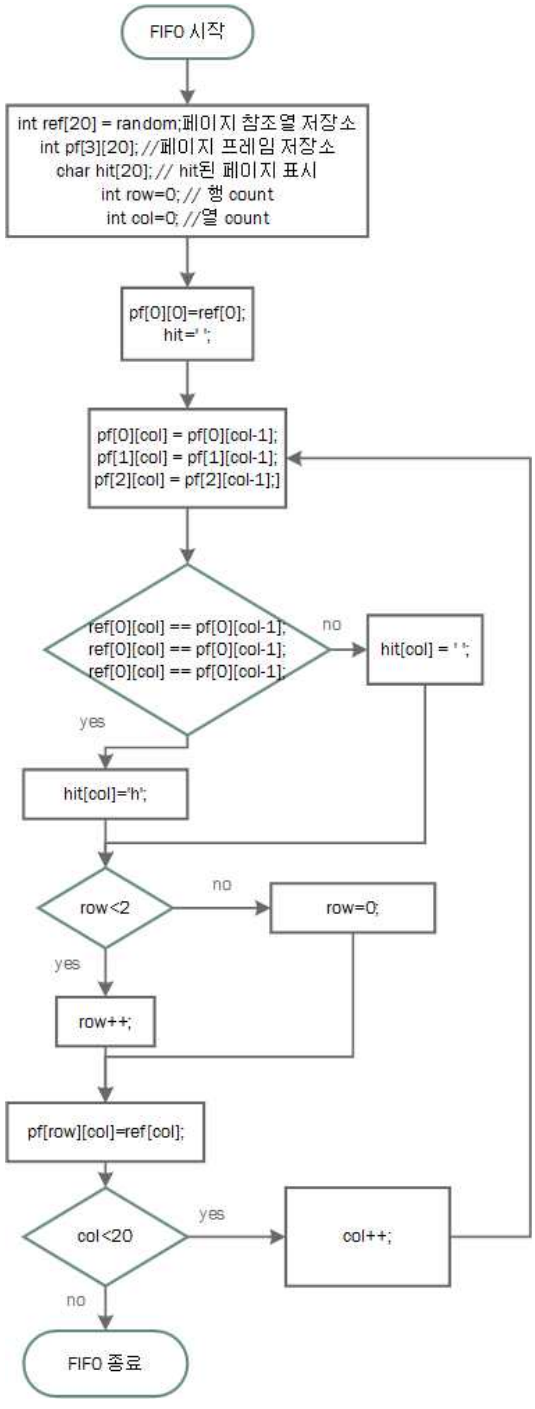

그림 3.13 프로그램의 FIFO 부분의 알고리즘

```
void setup() {
  size(800, 480);
  noStroke();
  smooth();
  controlP5 = new ControlP5(this);
  myFont = createFont("Aparajita-48", 16);
  textFont(myFont);
  text("교체정책", 45, 125);
  text(message, 47, 116);
  for (int a=0; a<12; a++) {
    img[a] = loadImage("page_0" + a +".png");
  }
  qnum=0;

  fifo = new pageFrame();
  fifo.load_fifo();

  lru = new pageFrame();
  lru.load_lru();

  opt = new pageFrame();
  opt.load_opt();

  comp = new pageFrame();
  comp.load_compare();
}  // setup()
```

표.4 프로그래밍 코드 /setup() 함수: 조기 값들/

```
void draw() {
  background(255);
  if (thisStage == 0) {
    image(img[0], 0, 0);
  }
  else if (thisStage == 1) {
    image(img[1], 0, 0);
  }
  else if (thisStage == 2) {
    image(img[2], 0, 0);
  }
  else if (thisStage == 3) {
    image(img[3], 0, 0);
  }
  else if (thisStage == 4) {
    image(img[4], 0, 0);
  }
  else if (thisStage == 5) {
    image(img[5], 0, 0);
  }
  else if (thisStage == 6) {
    image(img[6], 0, 0);
  }
if (thisStage == 2) {..} // FIFO 선택
if (thisStage == 3) {..} // LRU 선택
if (thisStage == 4) {..} // OPT 선택
if (thisStage == 5) {..} // 〈비교〉 선택
if (thisStage == 6) {..} // 〈평가〉 선택
} // draw
```

표.5 프로그래밍 코드 /draw 함수, thisStage /

```
 if (thisStage == 2) {  // FIFO 선택
    // FIFO 페이지 참조 큐 데이터 출력
    int rect_x = 130;
    stroke(0);
    tLocation = 113;
    for (int a=0; a<20; a++) {
      tLocation = tLocation + 28;
      fill(0);
      text(fifo.ref[a], tLocation, 115);  // 페이지 참조 큐 데이터 시작 위치 :
143, 115
      noFill();
      rect(rect_x, 125, 28, -30);     // 페이지 참조 큐 사각형 : 130, 120
      rect_x = rect_x + 28;
    }
    tLocation = 113;  // 페이지 프레임 출력
    for (int a=0; a<fifo.btn_cnt ; a++) {
      tLocation = tLocation + 28;
      noStroke();
      fill(255, 0, 0);
      if (fifo.pf_r[a] != 0) ellipse(tLocation-28, fifo.pf_r[a], 5, 5);
      fill(0);
      text(fifo.pf[0][a], tLocation, 178);
      }
 fill(0);
    text(fifo.btn_cnt, 410, 300);            // 페이지 참조 횟수 출력
    text(fifo.btn_cnt  - fifo.h_cnt, 410, 325);  // 페이지 부재 횟수 출력
    text(fifo.change_cnt, 410, 350);        // 페이지 교체 횟수 출력
....
  }  // if (thisStage == 2)
```

표.6 프로그래밍 코드 /draw FIFO, FIFO 알고리즘/

```
if (thisStage == 3) {  // LRU 선택
    // LRU 페이지 참조 큐 데이터 출력
    int rect_x = 130;
    stroke(0);

    tLocation = 113;
    for (int a=0; a<20; a++) {
      tLocation = tLocation + 28;
      fill(0);
      text(lru.ref[a], tLocation, 115);// 페이지 참조 큐 데이터 시작 위치 : 143,
110
      noFill();
      rect(rect_x, 125, 28, -30);     // 페이지 참조 큐 사각형 : 130, 120
      rect_x = rect_x + 28;
    }
    tLocation = 113;
    for (int a=0; a<lru.btn_cnt; a++) {
      tLocation = tLocation + 28;
      noStroke();
      fill(255, 0, 0);
      if (lru.pf_r[a] != 0) ellipse(tLocation-28, lru.pf_r[a], 5, 5);
      fill(0);
      text(lru.pf[0][a], tLocation, 178);
      if (lru.pf[1][a]==0) {
        text("", tLocation, 208);
      }     ....
 fill(0);
    text(lru.btn_cnt, 410, 300);              // 페이지 참조 횟수 출력
    text(lru.btn_cnt - lru.h_cnt, 410, 325);  // 페이지 부재 횟수 출력
    text(lru.change_cnt, 410, 350);        // 페이지 교체 횟수 출력
}  // if (thisStage == 3) LRU 선택
```

표.7 프로그래밍 코드 /draw LRU, LRU 알고리즘/

```
if (thisStage == 4) {  // OPT 선택
    // OPT 페이지 참조 큐 데이터 출력
    int rect_x = 130;
    stroke(0);
    tLocation = 113;
    for (int a=0; a<20; a++) {
      tLocation = tLocation + 28;
      fill(0);
      text(opt.ref[a], tLocation, 115);// 페이지 참조 큐 데이터 시작 위치 :
143,100
      noFill();
      rect(rect_x, 125, 28, -30);     // 페이지 참조 큐 사각형 : 130, 120
      rect_x = rect_x + 28;
    }
    tLocation = 113;
    for (int a=0; a<opt.btn_cnt; a++) {
      tLocation = tLocation + 28;
      noStroke();
      fill(255, 0, 0);
      if (opt.pf_r[a] != 0) ellipse(tLocation-28, opt.pf_r[a], 5, 5);
      fill(0);
      text(opt.pf[0][a], tLocation, 178);
      if (opt.pf[1][a]==0) {
        text("", tLocation, 208);
      } ....
fill(0);
    text(opt.btn_cnt, 410, 300);              // 페이지 참조 횟수 출력
    text(opt.btn_cnt - opt.h_cnt, 410, 325);  // 페이지 부재 횟수 출력
    text(opt.change_cnt, 410, 350);         // 페이지 교체 횟수 출력
 }  // if (thisStage == 4) OPT 선택
```

표.8 프로그래밍 코드 /draw OPT, Optimal 알고리즘/

```
if (thisStage == 5) {  // 〈비교〉 선택
    int rect_x = 130;
    stroke(0);
    tLocation = 113;  // text의 x 좌표
    for (int a=0; a<20; a++) {
      tLocation = tLocation + 28;
      fill(0);
      text(comp.ref[a], tLocation, 115); // 페이지 참조 큐 데이터 시작 위치 :
143, 110
      noFill();
      rect(rect_x, 125, 28, -30);     // 페이지 참조 큐 사각형 : 130, 120
      rect_x = rect_x + 28;
    }
    tLocation = 115;  // text의 x 좌표
    for (int a=0; a<comp.btn_cnt; a++) {
      tLocation = tLocation + 28;
      noStroke();
      fill(255, 0, 0);
      if (comp.c_pf_r[0][a] != 0) ellipse(tLocation-25, comp.c_pf_r[0][a], 5, 5);
// dot 표시
      if (comp.c_pf_r[1][a] != 0) ellipse(tLocation-25, comp.c_pf_r[1][a], 5, 5);
// dot 표시
      if (comp.c_pf_r[2][a] != 0) ellipse(tLocation-25, comp.c_pf_r[2][a], 5, 5);
// dot 표시
      fill(0);
      text(comp.c_pf[0][0][a], tLocation, 165);  // fifo 페이지 프레임의 첫 줄
      text(comp.c_pf[1][0][a], tLocation, 272);  // lru 페이지 프레임의 첫 줄
      text(comp.c_pf[2][0][a], tLocation, 384);  // opt 페이지 프레임의 첫 줄
      ...
}  // if (thisStage == 5) compare 선택
```

표.9 프로그래밍 코드 /draw compare, 비교 선택/

```
if (thisStage == 6) {  // 〈평가〉 선택
  //qnum=0;
    if(qnum==0){
     // text("문제 1, 2, 3, 4, 5, 정답", 47, 116);
    int a=47;
      for(int i=0; i<15; i++) {
       if(i==3){
          textSize(24);
          text(message.charAt(3), 92, 116);
      } else
      textSize(16);
      text(message.charAt(i), a, 116);
      a+=15;
      }
 else if(qnum==1) {
      int a=47;
      for(int i=0; i<15; i++) {

       if(i==5){
          textSize(24);
          text(message.charAt(5), 122, 116);
      } else
      textSize(16);
      text(message.charAt(i), a, 116);
      a+=15;
      }
...
}  // draw()
```

표.10 프로그래밍 코드 /draw compare, 평가 선택/

```
class pageFrame {
  int[] ref = new int[20];  // FIFO 참조 데이터 저장
  int[] pf_r = new int[20];  // 페이지 프레임(PF1, PF2, PF3) text 시작 위치

  int[][] pf = new int[3][20];     // 페이지 프레임 저장소
  char[] hit = new char[20];  // FIFO 페이지 부재 열 표시 "h"
  char[] ch = new char[20];  // FIFO 페이지 교체 열 표시 'c'
  int[][][] c_pf = new int[3][3][20];  // 3개의 알고리즘 비교(compare) 페이지 프레임 저장소
  char[][] c_hit = new char[3][20];  // load_compare()에서 LRU의 hit 문자 저장
  char[][] c_ch = new char[3][20];  // load_compare()에서 OPT의 교체 열 표시
  int[][] c_pf_r = new int[3][20];  // 페이지 프레임(PF1, PF2, PF3) text 시작 위치
  int l_rep = 0;  //페이지 교체 카운터
  int o_rep = 0;  // 페이지 교체 카운터
  int[][] count = new int[3][20];  // LRU 알고리즘의 교체 카운터
  int[][] cl_count = new int[3][20]; // load_compare()에서 LRU의 교체 카운터
  int[][] o_count = new int[3][20];  // OPT 알고리즘의 교체 카운터
pageFrame() {
    for(int a=0; a<3; a++) {
      for(int b=0; b<20; b++) {
        count[a][b]=0;
      }...
  void load_fifo(){ ..}
  void load_lru(){ ..}
  void load_opt(){ ..}
  void load_compare(){ ..}
    }
```

표.11 프로그래밍 코드 /pageFrame class /

```
// FIFO 알고리즘 : 적재된 시간이 가장 오래된 페이지 프레임을 선택하는 방법
void load_fifo() {
     pf[0][0] = ref[0];
     hit[0] = ' ';
for(col=1; col<20; col++) {
       pf[0][col] = pf[0][col-1];
       pf[1][col] = pf[1][col-1];
       pf[2][col] = pf[2][col-1];

       if( (ref[col] == pf[0][col-1]) ||
            (ref[col] == pf[1][col-1]) ||
            (ref[col] == pf[2][col-1]) ) {
              hit[col] = 'h';
              if(ref[col] == pf[0][col-1]) pf_r[col] = 168;
              if(ref[col] == pf[1][col-1]) pf_r[col] = 198;
              if(ref[col] == pf[2][col-1]) pf_r[col] = 228;
            } else {
                  hit[col] = ' ';
                  if(row<2) {
                     row++;
                  } else {
                  row = 0;
                  }
              pf[row][col] = ref[col];
              if( (pf[0][col-1] != 0) && (pf[1][col-1] != 0) && (pf[2][col-1]
!= 0) ) {
                 ch[col] = 'c';
              }
            }
       }
  }  // load_fifo()
```

표.12 프로그래밍 코드 /pageFrame class: FIFO 알고리즘(load_fifo())/

```
void load_lru() {
    // LRU 알고리즘 기록 : 가장 오래전에 사용된 페이지 프레임을 선택하는
방법
   pf[0][0] = ref[0];
    hit[0] = ' ';
    count[0][0]++;
    for(col=1; col<20; col++) {
      pf[0][col] = pf[0][col-1];
      pf[1][col] = pf[1][col-1];
      pf[2][col] = pf[2][col-1];
      if( (pf[0][col-1] == 0) || (pf[1][col-1] == 0) || (pf[2][col-1] == 0) )
{             if( (ref[col] == pf[0][col-1]) ||
            (ref[col] == pf[1][col-1]) ||
            (ref[col] == pf[2][col-1]) ) { //equal?
                hit[col] = 'h';
              if(ref[col] == pf[0][col-1]) {
                pf_r[col] = 168;
                pf[0][col]=ref[col];
                count[0][col]=1;
               //count[0][col]=count[0][col-1]+1;
              }
              if(ref[col] == pf[1][col-1]) {
                pf_r[col] = 198;
                pf[1][col]=ref[col];
               //count[1][col]=count[1][col-1]+1;
               count[1][col]=1;
              }..
            } ...}// load_lru()
```

표.13 프로그래밍 코드 /pageFrame class: LRU 알고리즘(load_lru())/

```
// OPT 알고리즘 : 앞으로 가장 오랫동안 사용되지 않을 프레임을 선택하는 방법
  void load_opt() {
    pf[0][0] = ref[0];
    hit[0] = ' ';
    for(col=1; col<20; col++) {
      pf[0][col] = pf[0][col-1];
      pf[1][col] = pf[1][col-1];
      pf[2][col] = pf[2][col-1];
      if( (pf[0][col-1] == 0) || (pf[1][col-1] == 0) || (pf[2][col-1] == 0) )
{  //empty?
       if( (ref[col] == pf[0][col-1]) ||
           (ref[col] == pf[1][col-1]) ||
           (ref[col] == pf[2][col-1]) ) { //equal?
             hit[col] = 'h';
           if(ref[col] == pf[0][col-1]) {
             pf_r[col] = 168;
             pf[0][col]=ref[col];
            }
           if(ref[col] == pf[1][col-1]) {
             pf_r[col] = 198;
             pf[1][col]=ref[col];
            } ...
           if( (pf[0][col-1] != 0) && (pf[1][col-1] != 0) && (pf[2][col-1]
!= 0) ) {
             ch[col] = 'c';
           }
         }
```

표.14 프로그래밍 코드 /pageFrame class: Optimal 알고리즘(load_opt())/

```
void load_compare() {
    c_pf[0][0][0] = ref[0];
    c_pf[1][0][0] = ref[0];
    c_pf[2][0][0] = ref[0];
    c_hit[0][0] = ' ';      // fifo
    c_hit[1][0] = ' ';      // lru
    c_hit[2][0] = ' ';      // opt
    cl_count[0][0]=1;

    for(col=1; col<20; col++) {
      c_pf[0][0][col] = c_pf[0][0][col-1];
      c_pf[0][1][col] = c_pf[0][1][col-1];
      c_pf[0][2][col] = c_pf[0][2][col-1];

      c_pf[1][0][col] = c_pf[1][0][col-1];
      c_pf[1][1][col] = c_pf[1][1][col-1];
      c_pf[1][2][col] = c_pf[1][2][col-1];

      c_pf[2][0][col] = c_pf[2][0][col-1];
      c_pf[2][1][col] = c_pf[2][1][col-1];
      c_pf[2][2][col] = c_pf[2][2][col-1];
     // Compare: FIFO 페이지 프레임 데이터 생성
     // Compare: LRU 페이지 프레임 데이터 생성
      // Compare: OPT 페이지 프레임 데이터 생성
       ...
}
}
```

표.15 프로그래밍 코드 /pageFrame class: 비교(load_compare())/

```
// Compare: FIFO 페이지 프레임 데이터 생성
        if( (ref[col] == c_pf[0][0][col-1]) ||
            (ref[col] == c_pf[0][1][col-1]) ||
            (ref[col] == c_pf[0][2][col-1]) ) {
              c_hit[0][col] = 'h';
              if(ref[col] == c_pf[0][0][col-1]) c_pf_r[0][col] = 152;    //
compare fifo의 dot 입력 y 좌표
              if(ref[col] == c_pf[0][1][col-1]) c_pf_r[0][col] = 178;
              if(ref[col] == c_pf[0][2][col-1]) c_pf_r[0][col] = 206;
            } else {
                  c_hit[0][col] = ' ';
                  if(row<2) {
                    row++;
                  } else {
                  row = 0;
                  }
              c_pf[0][row][col] = ref[col];
              if( (c_pf[0][0][col-1] != 0) && (c_pf[0][1][col-1] != 0) &&
(c_pf[0][2][col-1] != 0) ) {  // 페이지 교체
                 c_ch[0][col] = 'c';
              }
            }
```

표.16 프로그래밍 코드 /pageFrame class: 비교(load_compare())/ cont.

```
// Compare: LRU 페이지 프레임 데이터 생성
      if(  (c_pf[1][0][col-1]  ==  0)  ||  (c_pf[1][1][col-1]  ==  0)  ||
(c_pf[1][2][col-1] == 0) ) { // empty?

        if( (ref[col] == c_pf[1][0][col-1]) ||
            (ref[col] == c_pf[1][1][col-1]) ||
            (ref[col] == c_pf[1][2][col-1]) ) { //equal?
              c_hit[1][col] = 'h';
             if(ref[col] == c_pf[1][0][col-1]) {
               c_pf_r[1][col] = 258;
               c_pf[1][0][col]=ref[col];
              //cl_count[0][col]=cl_count[0][col-1]+1;
              cl_count[0][col]=1;
             }
             if(ref[col] == c_pf[1][1][col-1]) {
               c_pf_r[1][col] = 286;
               c_pf[1][1][col]=ref[col];
              //cl_count[1][col]=cl_count[1][col-1]+1;
              cl_count[1][col]=1;
             }
             if(ref[col] == c_pf[1][2][col-1]) {
               c_pf_r[1][col] = 316;
               c_pf[1][2][col]=ref[col];
              //cl_count[2][col]=cl_count[2][col-1]+1;
              cl_count[2][col]=1;
             }
           }        ...
          }
```

표.17 프로그래밍 코드 /pageFrame class: 비교(load_compare())/ cont.

```
// Compare: OPT 페이지 프레임 데이터 생성
      if(  (c_pf[2][0][col-1]  ==  0)  ||  (c_pf[2][1][col-1]  ==  0)  ||
(c_pf[2][2][col-1] == 0) ) { //empty?
       if( (ref[col] == c_pf[2][0][col-1]) ||
           (ref[col] == c_pf[2][1][col-1]) ||
           (ref[col] == c_pf[2][2][col-1]) ) { //equal?
              c_hit[2][col] = 'h';
             if(ref[col] == c_pf[2][0][col-1]) {
               c_pf_r[2][col] = 368;
               c_pf[2][0][col]=ref[col];
              }
             if(ref[col] == c_pf[2][1][col-1]) {
               c_pf_r[2][col] = 396;
               c_pf[2][1][col]=ref[col];
              }
             if(ref[col] == c_pf[2][2][col-1]) {
               c_pf_r[2][col] = 425;
               c_pf[2][2][col]=ref[col];
              }
           } else { //Adilhan bish
                 c_hit[2][col] = ' ';
                 if(c_opt_row<2) {
                   c_opt_row++;
                   //count[row][col]=count[0][col-1]+1;
                 } else {
                 c_opt_row = 0;
                 }
             c_pf[2][c_opt_row][col] = ref[col];
             }
```

표.18 프로그래밍 코드 /pageFrame class: 비교(load_compare())/ cont.

3.5 결론 및 향후 연구

본 논문에서 제안한 페이징 기법 시뮬레이션은 운영체제를 배우는 학습자들이 안드로이드 플랫폼에서 쉽게 다운로드 받아 사용할 수 있는 앱으로 컴퓨터 분야의 주요 과목인 운영체제를 학습에 도움을 주고자 개발하였다.

향후 더 많은 운영체제 관련 학습 내용을 시뮬레이션화하여 교육용 스마트 App.으로 개발한다면 운영체제 교수자와 학습자에게 유용한 학습도구로 활용될 수 있을 것이다.

참 고 문 헌

1. 국내문헌

1. 김근명. (2000). “학습자의 인지 양식에 따라 컴퓨터 시뮬레이션의 충실도 수준이 학업 성취에 미치는 효과”, 한국교원대 대학원.

2. 임진숙. (2001). “컴퓨터 구조 학습을 위한 시뮬레이션형 웹 코스웨어의 설계 및 구현”, 한국교원대 대학원.

3. 이숙회.(2000). “플래시와 자바 시뮬레이터를 이용한 컴퓨터 가상 메모리 학습프로그램”, 연세대.

4.. 정성균, 이상곤. (2011). “CPU 스케줄링을 학습하는 운영체제 시뮬레이션 프로그램의 설계 및 구현”, 한국멀티미디어학회, 제14권, 3호, pp449-461.

5. Silberschatz, P. Galvin, G. Gagne. (2010). Operating System Concept with Java 8thedition,Willy.

6. 다니엘 쉬프만, 랜덤웍스. 2011. 프로세싱, 날개를 달다 Learning processing.

7. 이영학, 이영호, 정욱진, 고주영, 이광호, 시재창. (2011). 재미있는 프로세싱.

8. 개념 이해를 위한 운영체제, 이성락. (2002). 제6장 가상 메모리 관리.

9. 재미삼아 프로세싱. (2011). 이영학, 이영호, 정욱진, 고주영, 이광호, 심재창.

10. Andrew.S Tahenbum. (2009). 운영체제론 Modern Operating Systems, Third Edition, 노삼혁, 이동희, 전홍석, 최종무 공역.

11. 김재천, 부재율, 소경희, 채선희, 예비· (2005). 현직 교사를 위한 교육과정과 교육평가, 제3판, 교육과 학사, 경기도.

기타

1. http://wiki.processing.org/w/Android
2. http://www.cs.uic.edu
3. http://www.processing.org
4. http://www.en.wikipedia.org
5. http://developer.android.com/index.html

ABSTRACT

Design and Implementation for Paging Techniques Simulator in Android-based

Gurragchaa, G
Major in Computer Engineering
Dept. of Computer Engineering
Graduate School, Hansung University

Operating System is a course which handles complex and abstract concepts related to its components and how an operating system works. However, most of the OS courses have been textbook-oriented theoretical classes. Therefore, many instructors have tried to make use of educational tools to help students understand a lecture and arouse their interests consistently. This paper describes the design and implementation of an Android-based page replacement simulator which shows the process of page replacement algorithms visually.

This simulator is easy to use for students because it designed as android application. This study helps the learner to clearly understand algorithms through the personal animation of actual embodiment process, to study the basic principles of the Operating System and, to build a solid foundation for him or herself as a computer expert.