게임 프로그래밍을 위한 순수 함수형 언어의 활용

Apply A Pure Functional Language for Game Programming

이동주, 변석우*, 우균
부산대학교 컴퓨터 공학과, 경성대학교 컴퓨터 과학과*

Lee Dong-Ju, Byun Suk-Woo*, Woo Gyun
Department of Computer Engineering, Pusan National University
Department of Computer Science, Kyungsung University*

요약

일반적으로 게임 프로그램을 구현할 때 C언어와 같은 명령형 프로그래밍 언어가 주로 사용된다. 하지만 복잡하고 다양한 그림의 모습 및 동작을 표현하기에는 프로그래머의 많은 노력이 요구된다. 복잡한 게임 프로그램을 보다 간결하고 명확하게 표현하는 방법으로 순수 함수형 언어인 Haskell을 이용하는 방법을 시도한다. Haksell과 같은 순수 함수형 프로그래밍은 기술적인 우수성을 가지고 있으며, 현재 많은 분야에서 특화된 언어로 되고 있다. 본 논문에서는 동일한 게임을 Haskell과 C로 직접 구현함으로서, 게임 개발 측면에서 두 언어의 차이점과 장단점에 대해서 논의한다. 또한 향후 Haskell의 실용화 가능성에 대해서도 논의한다.

Abstract

The imperative programming language like C language is Generally used when we develop the game program. But there is the need of much effort and time on low-level-details in order to express the game program that has complicated and varied motion. I will try to take measures to use the pure functional language, Haskell as the method of simplifying complex game program. The pure functional programming language like Haskell has excellence of technology and it has become specific in many domains. In this paper I'll discuss the difference between The two languages and merits and demerits in the game development aspect. Also discuss the possibility of putting Haskell to practical use in the future.

I. 서 론

최근 하드웨어의 눈부신 성장으로 인하여 개인용 컴퓨터에서 멀티미디어 환경은 표준이 되었다. 멀티미디어 환경을 위한 다양한 응용프로그램이 사용자에게 제공 되며 그중 게임 프로그램은 멀티미디어 관련 응용 프로그램 중 대표적인 응용 프로그램으로 볼 수 있다. 하지만 게임 프로그램을 개발하는 것은 쉬운 문제가 아니다. 예를 들어 게임 프로그램의 모습 및 동작을 표현하기 위해서는 하부 구조의 구체적인 관련부분들을(low-level details) 모두 일일이 표현해야 하며 이는 매우 단조롭고 지루한 작업이며 많은 노력이 소요된다.

일반적으로 게임 구현에는 C언어와 같은 명령형 프로그래밍 언어가 주로 사용이 된다. C언어와 같은

명령형 프로그래밍 언어를 이용하여 게임을 개발할 경우 점점 복잡해지는 게임 알고리즘, 화려한 그림 및 동작을 표현하는 방법은 매우 복잡하다. 그 이유는 명령형 프로그램의 특성상 모든 계산은 순차적으로 프로그래머가 명시하기 때문이며 그 게임의 복잡성과 프로그래머의 노력이 비례하기 때문이다. 따라서 이에 대한 대안으로 기존의 명령형 프로그래밍과는 다양한 대조를 이루는 순수 함수형 언어를 이용하여 게임을 구현하는 방법을 제시한다.

Haskell과 같은 순수 함수형 프로그래밍은 고급 수준의 추상화 기법과 타입 표현 능력의 장점을 갖고 있으며, 여러 논문과 연구를 통하여 그 기술적 우수성을 인정받아 왔다 [1][2].

본 연구에서는 순수 함수형 언어인 Haskell과 대표적인 명령형 프로그래밍 언어인 C를 이용하여 같은 알고리즘을 사용하는 테트리스(Tetris) 게임을 구현하였다. 또한 게임 구성요소 중 가장 핵심적으로 사용되는 요소를 선정하여, 각 언어별로 핵심요소의 구현 방법과 특징을 비교 분석하였다. 따라서 비교 분석한 결과를 토대로 앞서 제시한대로 게임 개발 시 순수 함수형 언어인 Haskell의 사용이 기존의 명령형 프로그래밍 사용의 대안이 되는지를 점검하려 한다.

본 논문은 다음과 같이 구성되어 있다. 먼저 2장에 게임에서 가장 핵심이 되는 비교 항목을 선정한 다음, 3장에서 각 언어별로 게임을 구현하여 비교한다. 4장에서 결과 분석을 통하여 게임 프로그램 영역에서 순수 함수형 언어의 적합성을 점검한다.

Ⅱ. 비교 항목 선정

1. 게임 데이터 표현

게임 프로그램에서 등장하는 모델이며, 논리적인 데이터로 나타내는 자료 구조로 볼 수 있다. 실제 게임 프로그래밍에서 가장 중요한 역할을 차지하며, 게임에 등장하는 모델이 복잡한 성격이나 다양한 제약을 가질 경우, 보다 복잡한 자료 구조를 가지게 된다. 또한 게임 데이터의 자료 구조에 따라 게임 데이터에 대한 연산의 복잡성도 결정되므로 게임 프로그램을 구성하는데 아주 중요한 요소로 볼 수 있다. 따라서 프로그래밍 언어 측면에서 제공되는 타입 및 사용자 정의 데이터 타입이 얼마나 게임 데이터 표현에 유용한지를 비교한다.

2. 게임 데이터 연산

게임 데이터는 각 게임 방법에 의해 정해진 조건 및 사용자 입력에 따라 변하게 된다. 따라서 게임 방법에 따른 복잡한 법칙을 만족하기 위해서는 게임 데이터를 조작하는 다양한 연산자가 필요하게 된다. 게임 데이터가 특정 조건에 만족하는지를 알아보는 연산자와 사용자의 입력이나 게임의 조건에 의해 게임 데이터의 값을 바꾸는 연산자로 크게 두 가지가 있다. 이와 같은 게임 데이터 연산자 구성을 위해 프로그래밍 언어 측면에서 제공되는 기본적인 데이터 연산자와 연산자를 이용하는 방법 등이 얼마나 게임 데이터 연산자를 구성하는데 유용한지를 비교한다.

3. 게임 흐름 제어

모든 종류의 게임 프로그램은 게임 순서를 가진다. 게임의 시작이 있으며, 반복적인 수행 중 특정 게임 조건을 만족하면 게임이 종료하게 된다. 반복적인 게임 흐름을 제어하는 것은 게임 프로그램의 메인 모듈에 해당한다. 따라서 프로그래밍 언어에서 제공되는 프로그램 흐름 제어 방법이 어떻게 게임 흐름제어에 적용되는지를 비교 분석한다.

Ⅲ. 테트리스 게임 구현

1. 게임 데이터 표현

테트리스는 게임 데이터를 나태나는 블록과 배경(Backgroud) 두 가지 구성 요소가 있다. 이중 블록은 핵심적인 게임 데이터이며 마우스나 키보드의 제

어를 받으면서 동작하는 reactive 및 동영상의 특성을 갖고 있는 객체이다. 상, 하, 좌, 우 등과 같이 4가지 입력을 받으며 하, 좌, 우는 해당방향으로 이동하며, '상'에 해당하는 키는 블록의 모양이 90도 회전시킨 모양으로 변한다. 총7가지의 종류의 블록이 있으며, 추가적으로 임의의 블록 모양을 만들 수 도 있다.

C : typedef을 사용하여 구조체 타입의 새로운 블록타입을 설정하였다. 블록 모양은 2차원 배열을 표현하였으며, 블록의 좌표 값은 Int형 변수 x와 y로 표현하였다.

```
typedef struct
{ BOOL *type;
  int x,y;
}BLOCK;
```

Haskell : Haskell에서는 type 키워드를 사용하여 기존에 정의된 타입에 새로운 타입이름을 붙일 수 있다 (C의 typedef과 유사). type 키워드를 사용하여 3가지 원소를 가지는 튜플 타입의 새로운 Block 타입을 설정하였다. 아래 블록의 모양은 Haskell의 리스트 (list)로서 표현하였으며, Boolean 타입을 사용하여 블록의 모양을 표현하였다.

```
type Block = ([[Bool]],Int,Int)
```

2. 게임 데이터 연산

- Mapping : 블록과 배경을 결합시켜 새로운 배경을 만든다.
- Processing : 유저의 입력(이벤트)으로부터 블록을 이동시키거나 회전시킨다.
- Turning : 현 블록의 모양을 시계 반대 방향으로 90도 회전한다.
- Checking : 현 블록의 위치에서 입력된 방향으로 회전의 가능여부를 확인한다.
- Deleting : 현재 배경의 모양에서 어떤 한 줄에 해당되는 블록이 다 찾을 경우 해당 라인의 블록이 삭제되고, 그 위에 있는 블록들은 삭제된 라인 수만큼 아래로 이동시킨다.
- Drawing : 현재 배경을 해당 시스템에 맞는 그래픽 타입으로 변형한다.

테트리스 게임을 구현하기 위해서는 위와 같은 기능을 하는 연산자가 꼭 필요하다. 위의 모듈들은 각각 C와 Haskell을 이용하여 구현되었는데, 그 중에서 게임 데이터가 특정 조건에 만족하는지를 알아보고 해당조건일 경우 그 게임 데이터 값을 수정 하는 연산자인 Checking 연산자를 C와 Haskell 코드를 소개하기로 한다.

Checking 및 Deleting 알고리즘 : 블록이 가로로 한줄 다 찾을 경우를 체크한다. 예를 들어, 배경이 아래와 같이 n번째 라인에 모든 블록이 다 찼을 경우를 가정하자.

line-1	True	False	....	False
... ...				
line-n	True	True	True	True

이 경우에 적용되어야 할 연산은 n번째 라인을 삭제하고, 라인 1부터 (n-1) 사이에 위치한 라인을 하나씩 밑으로 이동 (shift) 시키는 것이다.

```
C : void checkBack(BYTE* source,BYTE* dest) {
  ....
  for(i=0;i<20;i++) {
    cnt=cnt-2; cnt2=cnt2-2;
    for(j=0;j<12;j++) {
      test_true = (test_true&&*(source+cnt));
      *(dest+cnt2)=*(source+cnt);
      cnt--;cnt2--;}
     .....}
  .....}
Haskell : breaking :: Background -> Background
breaking b =
```

(filter filterF [upperProcess line | line <- init (init b)]) ++
(filter filterF [lowerProcess line | line <- init (init b)]) ++ [last b,last b]

C 코드는 배열의 인덱스 관리 등의 구현상의 구체적인 상황을 프로그래머가 일일이 모두 표현해야 하는데 비하여, Haskell 코드에서는 리스트 컴프리핸션(comprehension)의 기능과 고차 함수인 filter를 이용하여 프로그램의 코드를 매우 간결하고 명료하게 표현하고 있음을 알 수 있다.

3. 게임 흐름 제어

테트리스 게임 프로그램의 메인 모듈에 해당되며, 다음과 같은 과정을 순차적으로 반복하게 된다.

- 임의의 블록을 생성
- 입력 된 키에 맞게 블록 제어
- 배경과 제어된 블록의 매칭
- 매칭된 Map을 Draw
- 배경 체크

C :

```
if(!blockMove(MAP,DOWN,block)) {
    checkBack(SHOWMAP,MAP);
  blockSelect(block);}
else {
    blockToMap(MAP,block, SHOWMAP);
    drawMap((BYTE*)SHOWMAP,hdc);}
```

Haskell :

```
loop (block,ground,over1) = do
...
(newBlock,nextOk) <- processingBlock...
case nextOk of
        False -> gameloop ((checkingBackground..))
        True -> return()
...
loop (newBlock,ground,True)
```

C 프로그래밍에서 어떤 일들의 순차적인 표현은 문장들을 ';'로 연결함으로써 쉽게 표현할 수 있다. Haskell 프로그래밍에서는 이에 대응하는 do 모나드를 이용하여 표현할 수 있다. 위에서 C로 작성된 순차적 반복 진행과정은 do와 재귀적인 함수 정의를 이용하여 마치 명령형 프로그래밍 언어에서 사용하는 방법과 유사하게 표현할 수 있다.

Ⅳ. 결과 분석

게임 데이터 표현에 있어 C는 구조체를 이용한 사용자 정의 타입 구성하였으며, 블록 데이터 구성하기 위해서 C에서 제공하는 2차원 배열을 사용했다. 게임 데이터 연산에 있어 구조체로 정의된 게임 데이터의 조작을 위해서 데이터가 있는 각 배열을 반복문을 사용하여 값을 검사하거나, 값을 조작한다. 따라서 저장된 배열의 인덱스 관리 등의 구현상의 구체적인 상황을 모두 표현해야 되며, 프로그래머에게 복잡한 표현을 요구한다. 이와 같은 현상은 C에서는 배열에 대한 다양한 연산자를 제공하지 않으며, 프로그래머가 모든 상황을 구현해야 하는 이유로 볼 수 있다. 반면 게임 흐름 제어 모듈을 구현할 때 C는 아주 자연스럽고 적합하게 표현이 가능하다. C 언어에서는 어떤 일들의 순차적인 표현은 문장들을 ';'로 연결함으로써 쉽게 표현이 가능하다. 또한 반복적인 순차 구조를 표현할 경우, C언어에서 제공하는 반복문을 적절히 이용하면 아주 쉽게 표현이 가능하다.

함수형 언어인 Haskell에서는 다형 타입인 튜플(Tuple)을 이용하여 사용자 정의 타입 구성하였으며, 블록 데이터 구성하기 위해서 다형 타입인 리스트를 사용하였다. 블록 데이터를 조작하기 위해서는 리스트 연산이 필요하다. 리스트 연산 시 리스트 컴프리핸션과 고차 함수를 이용하면 리스트의 인덱스 관리

[표 1] C와 Haskell에서 지원되는 언어적 기능 비교

언어적 기능	C	Haskell
단형 타입	O	O
다형 타입	X	O
컴프리헨션	X	O
고차함수	X	O
순차구조	O	O†
선택구조	O	O
반복구조	O	O‡
상태사용	O	O□□

†) 모나드(Monad)를 이용하여 순차구조를 지원
‡) 함수의 재귀 호출을 이용하여 반복 구조를 지원
□□) 상태 모나드(State Monad)를 이용하여 상태 사용을 지원

와 같은 구체적인 상황을 기술할 필요 없는 특징이 있다. 따라서 리스트 컴프리헨션과 고차 함수를 이용함으로서 간단하고 명료하게 게임 데이터 연산자를 구성하였다. Haskell은 순차적인 계산과정을 명시하는 방법으로 모나드를 이용한다. Haskell의 기본 라이브러리로 제공되며, 문법적으로 Do 표기법을 제공함으로써 순차적인 계산과정을 마치 C 언어에서 표현 방법과 유사한 표현 방법을 지원 해준다. 또한 반복적인 순차 구조를 표현할 경우, 재귀함수 과 모나드를 적절히 이용하면 이와 같은 표현은 간단히 해결할 수 있었다.

앞서 게임의 핵심 세 가지 요소를 C 와 Haskell를 이용하여 비교 분석해 보았으며 그 결과에 따라 다음과 같은 견해를 갖게 되었다.

첫째, 게임의 논리적인 부분의 핵심이 되는 '게임 데이터 표현' 과 '게임 데이터 연산'이라는 이 두 가지 항목에서는 함수형 언어인 Haskell이 C언어에 비해 여러 가지 장점을 내포하고 있다. 기존의 C언어에 비해 Haskell의 높은 추상적 표현 능력에 따라 프로그램의 표현이 C언어 보다 훨씬 간결하게 표현될 수 있음을 확인할 수 있었다. 같은 알고리즘과 비슷한 환경을 이용하여 개발된 테트리스 게임에 있어서, C를 이용하여 500 라인의 코딩을 하였으나, Haskell을 이용할 경우 100 라인으로 코딩할 수 있었다. 전반적으로 볼 때 Haskell 코드는 C 코드의 약 1/5 정도로 압축하여 프로그래밍 할 수 있었다. 일반적으로 Haskell의 코드는 C에 비해 1/10에서 1/20 정도인 것으로 알려 졌는데, 1/5 인 것은 프로그램의 규모가 작기 때문인 것으로 사료되며, 좀 더 최적화된 코딩을 적용하면 그 양은 더욱 줄어들 것으로 예상한다. 코드의 양을 줄일 수 있었던 결정적인 이유는 Haskell의 강력한 리스트 컴프리헨션 기능과 고차 함수 때문인 것으로 파악되고 있다.

둘째, 순수 함수형 언어인 Haskell에서는 순차적인 계산 과정을 표현하는데 많은 어려움이 있었으나 현재 Haskell98에서는 모나드를 라이브러리 형태로 제공해 줌으로서 이 문제는 쉽게 해결이 된다. 따라서 게임 프로그래밍 등에서 처리해야 하는 계산과정의 순차성을 표현하는 문제와 관련하여 Haskell의 do는 비교적 수월하게 표현될 수 있었으며, 특히 (x, y) <- 과 같이 튜플로 된 변수들의 값을 할당은 아주 쉽게 표현할 수 있었다.

셋째, 게임 등의 멀티미디어 소프트웨어를 제작하기 위해서는 음성, 그래픽, 동영상 등의 다양한 유틸리티들을 이용하는 하는 것이 필요하다. Microsoft 사의 Visual C는 이 분야에 있어서 강력한 환경을 제공하고 있는 것에 비하여, 본 연구에서 이용한 Haskell 인터프리터 Hugs는 충분한 유틸리티들을 제공하지 못하고 있다. 그럼에도 불구하고 기본적인 유틸리티들을 제공하고 있으며 테트리스 게임을 구현하기에는 충분하였다. 그러나 Hugs로 개발된 프로그램을 Haskell 컴파일러인 GHC를 이용하여 수행파일을 만들려고 할 때 이 두 언어가 사용하는 유틸리티들이 상이한 특징 때문에 프로그램을 그대로 이용할 수 없는 문제점들이 발생하였다.

V. 결 론

기술적으로 많은 주목을 받고 있는 순수 함수형 언어 Haskell와 아주 일반적으로 사용되는 C를 이용하여 테트리스 게임을 개발하였다. 이 과정을 통해 기

존의 C언어로 구현한 게임 프로그램에 비해 Haskell로 구현한 게임 프로그램이 보다 우수한 생산성 과 간결한 코드로 좋은 가독성을 확인할 수 있었다. 또한, 계산과정을 명시적으로 표현하는 문제에 있어서, 모나드 기법에 기반한 Haskell의 do는 비교적 쉽고 자연스러운 순차적 표현을 가능케 하고 있다.

Haskell을 상용 프로그램 개발 언어로서 사용하기 위해서는 라이브러리와 유틸리티와 관련하여 더 많은 환경 및 표준화를 이룩하는 것이 필요하다고 보여진다. 현재 GHC와 Hugs에서 사용되는 기본 유틸리티들은 아직 호환성을 갖지 못하고 있다. 현재 이 분야에 많은 연구가 진행되고 있으므로 향후 연구를 기대해 본다.

테트리스 게임은 프로그래밍의 패턴이 비교적 단순하다. 향후 테트리스보다 프로그래밍의 패턴이 복잡한 형태의 게임을 개발해 봄으로써 게임 분야에 있어서 Haskell 프로그래밍의 실용성을 점검하는 일이 필요하다고 보여 진다. 테트리스 개발 경험을 미루어 볼 때, 유틸리티 분야를 좀 더 풍부하게 이용할 수 있다면 복잡한 게임 프로그램을 개발하는 것도 어렵지 않게 해결할 수 있을 것으로 전망한다.

결론적으로 게임분야에 있어 Haskell 프로그래밍은 매우 흥미로우며 향후 실용화 가능성도 높다고 보여 진다. 국내에서도 순수 함수형 프로그래밍 분야에 대한 연구와 노력이 적은 실정이다. 기술적으로나 경제적인 측면을 고려할 때, 향후 이 분야의 관심을 갖고 노력 하는 것이 필요할 것이다.

▌참 고 문 헌▐

[1] Haskell home page. http://www.haskell.org/

[2] John Hughes. Why functional programming mattes. Computer Journal, 32(2):98-107, 1989.

[3] E Moggi. Computational lambda-calculus and Monads. In Proceedings of Symposium on Logic in Computer Science. IEEE. 1989.

[4] P. Wadler. Comprehending Monads. In Proceedings of Symposium on Lisp and Functional Programming, pp. 61-78. ACM. June 1990.

[5] P. Hudak. The Haskell School of Expression: Learning Functional Programming Through Multimedia. Cambridge University Press, March 2000.

[6] Simon Thompson, Haskell: The Craft of Functional Programming. Addison-Wesley. 1996.