
테트리스 게임도
만들면서 배우는
나의
프로그래밍

PROGRAM WA KOSHITE TSUKURARERU
PROGRAMMER NO ATAMA NO NAKA WO NOZOITEMIYO

by Takashi Hirayama

First published in Japan in 2013 by Shuwa System Co., Ltd.

This Korean edition is published by arrangement with Shuwa System Co., Ltd, Tokyo in care of Tuttle-Mori Agency, Inc. Tokyo through Danny Hong Agency, Seoul.

초판 1쇄 발행 2015년 9월 30일

지은이 히라야마 타카시
옮긴이 김완섭
펴낸이 장성두
펴낸곳 제이펍

출판신고 2009년 11월 10일 제406-2009-000087호
주소 경기도 파주시 문발로 141 뮤즈빌딩 403호
전화 070-8201-9010 / **팩스** 02-6280-0405
홈페이지 www.jpub.kr / **이메일** jeipub@gmail.com

편집부 이민숙, 이 슬, 이주원 / **소통·기획팀** 민지환, 현지환
용지 신승지류유통 / **인쇄** 한승인쇄 / **제본** 광우제책사

ISBN 979-11-85890-31-9 (93000)
값 25,000원

테트리스 게임을 만들면서 배우는

나의 첫 프로그래밍

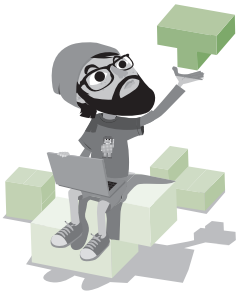
히라야마 타카시 지음 / 김완섭 옮김

Jpub 제이펍

※ 드리는 말씀

- 이 책에 기재된 내용을 기반으로 한 운용 결과에 대해 저자, 역자, 소프트웨어 개발자 및 제공자, 제이펍 출판사는 일체의 책임을 지지 않으므로 양해 바랍니다.
- 이 책에 등장하는 각 회사명, 제품명은 일반적으로 각 회사의 등록 상표 또는 상표입니다. 본문 중에는 ™, ©, ® 마크 등이 표시되어 있지 않습니다.
- 이 책에서 사용하고 있는 제품 버전은 독자의 학습 시점이나 환경에 따라 책의 내용과 다를 수 있습니다.
- 본문 중 일본 내의 실정에만 국한되어 있는 내용이나 그림은 일부를 삭제하거나 국내 실정에 맞도록 변경하였으니 참고 바랍니다.
- 책 내용과 관련된 문의사항은 옮긴이나 출판사로 연락해 주시기 바랍니다.
  - 옮긴이: jinsilto@gmail.com
  - 출판사: jeipub@gmail.com

차례



옮긴이 머리말 xiv
베타리더 후기 xvi

CHAPTER 0 이 책은 누구에게 무엇을 제공하는가? 1

0.1 기존 학습법의 세 가지 문제 2
0.1.1 요소가 제각각 2
0.1.2 전문가용 도구가 초래하는 좌절 3
0.1.3 이해하지 못해도 어쩔 수 없다 4
0.2 이 책은 무엇이 다른가? 5
0.2.1 하나의 프로그램을 처음부터 끝까지 만들어본다 5
0.2.2 전용 도구를 제공한다 6
0.2.3 정말로 처음인 사람을 대상으로 한다 7
0.3 그러면 시작해보자! 8

CHAPTER 1 프로그램을 만들기 전에 9

1.1 게임을 만들려면 어떻게 해야 할까? 9
1.1.1 게임에 필요한 프로그램 기술은 무엇인가? 10
1.1.2 이 책에서 말하는 게임이란? 10
1.1.3 프로그램이란 무엇인가? 16
1.1.4 이 해괴한 문장의 정체는? 17
1.1.5 여러분의 목표는 어느 정도의 수준인가? 18
1.2 무엇부터 시작할 것인가? 22
1.2.1 프로그래밍 언어 학습법 22
1.2.2 어떤 것부터 만들어야 할까? 23
1.2.3 만들기 쉬운 요소를 선별할 수는 없는가? 23
1.3 이번 장에서 알려주고 싶었던 것 24

CHAPTER 2 메모리 건드려보기 – 사각형 그리기 – 25

2.1 작성한 프로그램을 움직여보자 25

2.2 사각형 그리기 30

2.2.1 사각형을 그리기 위한 최소한의 규칙 30
2.2.2 이 한 줄은 무엇을 의미하는가? 31
2.2.3 메모리란 무엇인가? 33
2.2.4 메모리를 건드리면 점이 표시되는 이유는? 35
2.2.5 컴퓨터는 어떤 기계인가? 36
2.2.6 화면의 점은 몇 번 메모리에 연결돼 있는가? 37
2.2.7 메모리에 기억시키는 숫자와 색의 관계 41

2.3 원하는 위치에 사각형 그리기 44

2.3.1 화면 가운데에 빨간색 사각형 그리기 44
2.3.2 1을 더하거나 빼는 것이 귀찮다 48
2.3.3 왜 0부터 시작하는가? 50
2.3.4 이것으로 게임의 그림을 그릴 수 있는가? 54

2.4 이번 장에서 알려주고 싶었던 것 55

CHAPTER 3 반복 – 프로그램 짧게 만들기 – 57

3.1 행 수를 줄인다 58

3.1.1 한 행으로 여러 메모리를 건드리고 싶다 59
3.1.2 왜 반복인가? 60

3.2 반복 사용법 61

3.2.1 반복 문법 61
3.2.2 사각형을 그리기 위해 필요한 것은? 63
3.2.3 고정되지 않은 수 64
3.2.4 한 번 실행하고 끝낸다 65
3.2.5 두 번 실행하고 끝낸다 66
3.2.6 16회 실행한 후 끝내려면 어떻게 하면 될까? 69
3.2.7 원하는 만큼 반복하는 방법 70
3.2.8 반복할 때마다 다른 화소를 칠하려면 71

3.3 반복을 이용한 사각형 그리기 75

3.3.1 하지만 그렇게 간단하지 않다 75
3.3.2 반복을 반복하려면 어떻게 해야 하나? 77
3.3.3 이 프로그램의 어디가 잘못된 것일까? 80
3.3.4 머릿속에서 재현한다 81
3.3.5 결과부터 생각한다 83
3.3.6 생각한 대로 동작하지 않을 때 해야 할 것 88
3.3.7 반복을 이용해서 작성하는 것이 의미가 있는가? 90
3.3.8 화면 가운데 그리려면? 91

3.4 이번 장에서 알려주고 싶었던 것 92

CHAPTER 4 프로그램 변형 – 사각형을 많이 그리기 – 95

4.1 반복을 이용해서 벽 그리기 95
4.1.1 사각형 그리기를 반복한다 96
4.1.2 20개 그리기 99
4.1.3 이 분위기를 타고 오른쪽 벽과 바닥도 그려보자 102
4.1.4 주석 105
4.2 더 짧게 만들고 싶다 107
4.2.1 반복을 합친다 108
4.2.2 더 짧게 만들 수 있는 방법은? 111
4.2.3 바닥도 포함해서 짧게 작성하는 방법은 없을까? 112
4.2.4 이것은 알기 쉬운 프로그램일까? 117
4.3 이번 장에서 알려주고 싶었던 것 119

CHAPTER 5 프로그램을 나눈다 – 사각형을 쉽게 많이 그리기 – 121

5.1 코드 재사용을 위한 반복 이외의 수단 122
5.1.1 참조를 사용하면 어떻게 바뀔까? 122
5.1.2 이해하기 쉬운 참조 125
5.2 Sunaba의 '참조' 127
5.2.1 실행에 대해서 128
5.2.2 부분 프로그램이 도움이 되기 위해 필요한 것 130
5.2.3 참조할 때마다 다른 점을 그리고 싶다 130
5.3 부분 프로그램을 사용하여 벽과 바닥 그리기 131
5.3.1 좌우 벽도 부분 프로그램을 사용해서 그리자 133
5.3.2 부분 프로그램의 가치는? 138
5.4 더 알기 쉽도록 140
5.4.1 주석은 왜 필요한가? 141
5.4.2 '이상한 수'에 주목하자 142
5.5 이번 장에서 알려주고 싶었던 것 148

CHAPTER 6 움직임이 있는 프로그램 – 사각형 떨어뜨리기 – 151

6.1 무엇을 해야 하나? 152
6.2 사각형을 떨어뜨린다 153
6.2.1 점을 떨어뜨려 보자 154
6.2.2 사각형을 떨어뜨려 보자 156
6.2.3 틀리기 위한 조건 생각하기 160
6.2.4 문제를 수정한다 165

6.3 더 빨리 그리고 싶다 168
6.3.1 속도 저하를 해결하는 방법 169
6.3.2 속도 저하란 무엇인가? 170
6.3.3 편지 보내기 172
6.4 지금까지의 내용 반영하기 174
6.4.1 하나로 합치기 174
6.4.2 벽과 바닥 그리기 177
6.4.3 속도 저하 중지 178
6.4.4 떨어지는 사각형을 가운데로 178
6.5 이번 장에서 알려주고 싶었던 것 180

CHAPTER 7 메모리에 이름 붙이기 – 한글을 사용해서 – 181
7.1 현재 프로그램은 무엇이 문제인가? 182
7.1.1 어떤 도구가 필요한가? 184
7.1.2 번호를 직접 정하고 싶지 않다면? 186
7.2 Sunaba에서 메모리에 이름 붙이기 189
7.2.1 일단 사용해보자 190
7.2.2 메모리 번호를 정하지 않아도 된다는 것은? 191
7.3 이름 지정 메모리의 규칙 193
7.3.1 이름 지정 메모리가 만들어지는 것은 언제? 193
7.3.2 이름 지정 메모리는 부분 프로그램에 한정된다 194
7.3.3 반복 밖에서는 보이지 않는다 196
7.4 이번 장에서 알려주고 싶었던 것 197

CHAPTER 8 조작 가능한 프로그램 – 사각형 움직이기 – 199
8.1 조작한다는 것은 어떤 의미인가? 200
8.1.1 키보드에 반응한다 200
8.1.2 프로그램은 밖을 어떻게 인식할까? 200
8.2 조작할 수 있게 만들기 203
8.2.1 누를 틈이 없으며, 누르더라도 알 수 없다 204
8.2.2 움직이고 있는 동안 키를 누르고 싶다 205
8.2.3 점을 움직여보자 207
8.2.4 속도 저하 끄기 208
8.2.5 다른 방향으로도 움직이게 하기 209
8.2.6 사각형 움직이기 210

8.3 계속적인 움직임을 방지 212
8.3.1 작은 것부터 먼저 시작하기 213
8.3.2 계속 움직이지 않는다는 것은 무엇을 의미하나? 213
8.3.3 이전이 눌려 있지 않으면 어떻게 하는가? 215
8.3.4 메모판을 보고 어떻게 할지를 결정한다 217
8.3.5 식 생각해내기 219
8.3.6 상하좌우로 움직이기 222
8.3.7 사각형으로 바꾸자 224
8.4 떨어지는 사각형을 움직여보자 225
8.4.1 금방 할 수 있는 벽과 바닥 그리기 226
8.4.2 사각형 떨어뜨리기 227
8.5 이번 장에서 알려주고 싶었던 것 228

CHAPTER 9 '계산'의 진정한 의미 – 천천히 떨어뜨리기 – 231
9.1 천천히 떨어진다는 것 231
9.1.1 매번 움직이지 않는 프로그램 233
9.1.2 이전에 움직였으면 지금은 움직이지 않는다 234
9.1.3 나눗셈 사용하기 235
9.1.4 다른 한 가지 방법 237
9.2 조건 실행 238
9.2.1 특정 행의 실행 여부 조정 238
9.2.2 반복을 한 번만 한다 241
9.2.3 조건 실행은 유용한가? 242
9.3 Sunaba의 조건 실행 244
9.3.1 앞의 예는 어떻게 바뀔까? 245
9.3.2 천천히 선 그리기 245
9.3.3 현재까지의 프로그램 조합하기 248
9.3.4 계속 키를 누르고 있는 경우의 처리 250
9.4 계산의 진짜 의미 252
9.4.1 조건식이 성립한다는 것 253
9.4.2 <는 도대체 무엇인가? 254
9.4.3 Sunaba의 계산 기호 256
9.4.4 그럼, 사용해보자 257
9.4.5 사용해보자 260
9.5 이번 장에서 알려주고 싶었던 것 262

CHAPTER 10 메모리를 묶어서 사용하기 – 사각형 쌓아올리기 – 265

10.1 '접한다'는 것 266

10.1.1 바닥에 접한다는 것은 어떤 의미인가? 266
10.1.2 벽에도 접하게 해보자 271
10.1.3 다음으로 처리해야 할 것 273

10.2 메모리를 그룹으로 사용한다 275

10.2.1 문제를 변형한다 275
10.2.2 쌓인 순서대로 기억한다 277
10.2.3 모든 칸의 상태를 기억한다 279
10.2.4 두 가지 방법을 비교해보자 283

10.3 쌓아올리기 285

10.3.1 어떻게 할지 정리해보자 285
10.3.2 출발점 286
10.3.3 바닥에 접하면 쌓는다 286
10.3.4 새로운 사각형 등장시키기 288
10.3.5 바닥 외에도 쌓이도록 만들기 291
10.3.6 짧게 만들기 294
10.3.7 벽과 좌우 이동 기능 복원 297
10.3.8 프로그램 정리하기 300
10.3.9 쌓기 부분 프로그램 303
10.3.10 떨어지는 속도와 키 눌림 상태 대응 305

10.4 이번 장에서 알려주고 싶었던 것 308

CHAPTER 11 더 많은 메모리에 이름 붙이기 – 열 삭제하기 – 309

11.1 일단은 지우는 것부터 309

11.1.1 대략적인 순서 310
11.1.2 지우는 프로그램 311
11.1.3 조합하기 316
11.1.4 확인을 편하게 하고 싶다 318
11.1.5 떨어지지 않는 것을 확인해두기 320

11.2 프로그램 개선하기 322

11.2.1 번호 지정 메모리에 이름을 붙일 수 없다 322
11.2.2 부분 프로그램에서 다른 프로그램을 사용하는 경우 325
11.2.3 부분 프로그램에 정보를 전달하는 새로운 방법 327
11.2.4 입력 사용해보기 330
11.2.5 입력으로 문제를 해결할 수 있는가? 332
11.2.6 입력을 사용해서 전체 프로그램 수정 334

11.3 떨어뜨리기 ····· 336

11.3.1 가장 간단한 예 336
11.3.2 프로그램으로 만들기 338
11.3.3 위에 많은 사각형이 있는 경우 342
11.3.4 위에 있는 것 모두를 떨어뜨리기 344
11.3.5 다른 한 가지 방법 346
11.3.6 어느 쪽이 좋은가? 349

11.4 이번 장에서 알려주고 싶었던 것 ····· 350

CHAPTER 12 모든 것을 조합한다 – 사각형 회전하기 – 353

12.1 두 개로 늘리자 ····· 354

12.1.1 떨어지는 사각형은 어떻게 표현돼 있나? 355
12.1.2 두 개의 빨간색 사각형을 표시하는 단순한 방법 356

12.2 회전한다는 것은? ····· 360

12.2.1 회전 이외의 요소는 잠시 접어두자 360
12.2.2 결과부터 생각한다 362
12.2.3 회전의 네 가지 방식 364
12.2.4 좀 더 현명하게 작성하기 366

12.3 회전 기능을 구현하기 전에 ····· 370

12.3.1 두 개의 사각형을 그리는 프로그램 370
12.3.2 한 단계씩 정리하기 372
12.3.3 움직일 수 있는지 확인하는 방법 다시 생각하기 375
12.3.4 다른 방법 377
12.3.5 두 칸의 위치 기억하기를 중단하기 383
12.3.6 부분 프로그램을 더 활용하자 385
12.3.7 부분 프로그램의 출력 387
12.3.8 이제 정리는 끝! 389

12.4 회전 기능 추가하기 ····· 391

12.4.1 회전 기능이 추가되면 바뀌는 것 391
12.4.2 대략적으로 작성해보기 393
12.4.3 입력을 추가한 부분 프로그램 394
12.4.4 칸 위치 계산하기 397
12.4.5 드디어 회전! 398

12.5 제대로 동작하는가? ····· 400

12.5.1 이상하다! 401
12.5.2 어떻게 이상한가? 402
12.5.3 수정하기 404

12.6 이번 장에서 알려주고 싶었던 것 ····· 408

CHAPTER 13 '표'라는 생각 방식 – 게임의 완성 – 411

13.1 세 개씩 떨어뜨리기 411

13.1.1 프로그램상의 2와 3의 차이 412
13.1.2 세 개를 연결하는 방법은 하나만 있는 것이 아니다 414
13.1.3 프로그램으로 만들기 416
13.1.4 두 종류의 모양 사용하기 418
13.1.5 회전 기능도 넣자 421
13.1.6 '표'라는 생각 방식 423
13.1.7 표를 미리 만들어둔다 428

13.2 네 개씩 떨어뜨리기 430

13.2.1 세 칸과 네 칸의 차이 431
13.2.2 네 칸으로 만들 수 있는 수많은 모양 434
13.2.3 대표 사각형과 회전하기 전의 모양 정하기 436
13.2.4 프로그램으로 만들어보자 436
13.2.5 정말 이것으로 끝인가? 440
13.2.6 수고를 줄이기 위해서 441
13.2.7 표의 내용은 무엇인가? 441
13.2.8 회전 관련 표 줄이기 443
13.2.9 회전 관련 표를 더 줄이자 447

13.3 이번 장에서 알려주고 싶었던 것 454

CHAPTER 14 모래밭을 떠나자 – 끝이 시작이다 – 457

14.1 지금까지 사용했던 생각 방식 457

14.1.1 결과부터 생각하기 457
14.1.2 질문 방법 458
14.1.3 목표와 수단 460
14.1.4 선택 가능한 대안을 여러 개 준비한다 461
14.1.5 정리 462

14.2 더 진화하기 위해서는 462

14.2.1 Sunaba를 좀 더 가지고 놀아보자 463
14.2.2 실용 언어의 세계로 465

14.3 실용 언어가 어려운 이유 465

14.3.1 사용할 수 있는 상태로 만드는 것이 어렵다 465
14.3.2 실행하기 위한 순서 466
14.3.3 문법이 복잡하다 466
14.3.4 그림이나 소리를 만드는 것이 어렵다 466
14.3.5 객체지향 프로그래밍 467
14.3.6 영어 형식으로 작성해야 한다 468
14.3.7 메모리가 보이지 않는다 470
14.3.8 추천하고 싶은 것은 자바스크립트 470

APPENDIX A Sunaba 추가 사양 – 영어 버전 중심 – 473

A.1 메모리 변경 473
A.2 계산 473
A.3 반복과 조건 실행 474
A.4 이름 지정 메모리 474
A.5 부분 프로그램 475
A.6 메모리 번호 475
A.6.1 화면(60000번호)에 대해서 476

에필로그 477
찾아보기 479

옮긴이 머리말

요즘 IT 개발자들에게 가장 필요한 자질 중 하나가 '인문학적 소양'일 것이다. 단순히 기술적으로만 뛰어난 사람이 아니라 인문학적 방식으로 접근하고 생각할 수 있는 사람이 인재로 대우받는 세상이 온 것이다. 하지만 나 자신을 포함해서 대부분의 IT 개발자들은 단순명료한 것을 선호하는 편이다. 어떤 사양서나 기획서도 시스템이라는 형태로 만들어질 수 있을 정도로 분명하게 작성돼 있어야 하고, 또 그렇게 작성해야 한다. 이렇다 보니 자연스럽게 결여되는 것이 인문학적인 사고방식 또는 접근방식이다. 이 책이 가르치는 것은 방법 자체가 아니라 방법을 생각해낼 수 있는 능력이다. 따라서 단순한 프로그램 입문서가 아니라 인문학적 소양을 갖춘 개발자를 만들어주는 책이라고 말하고 싶다. 프로그램 입문자뿐만 아니라 기존 개발자에게도 새로운 시각으로 프로그래밍에 접근할 수 있는 기회를 주리라 생각한다.

개인적으로 이 책을 높게 평가하는 이유는 다음과 같다.

하나. 프로그래밍 언어가 한글로 돼 있다!
하나. 책을 다 읽고 나서는 감동을 받을 수 있다!
하나. 책을 읽어 나갈수록 더 읽고 싶어지는 책이다!
하나. 프로그램을 전혀 모르는 사람도 게임을 만들 수 있다!
하나. 프로그램을 아는 사람이라면 전혀 새로운 관점에서 접근할 수 있다!

이 책은 한 줄 한 줄 주옥같은 명언으로 가득 차 있는 책이다. 좋은 영화를 접하게 되면 몇 번이고 다시 영화를 보게 된다. 하지만 책의 경우 그런 책을 만나기는 쉽지 않다. 이 책은 매우 쉬우면서도 논리적이며, 인문학적인 책이다. 그러나 처음 읽을 때는 코드를 따라가기에 바쁠 수도 있다. 그러다 보면 이 책에 담겨 있는 주옥같은 명언들을 놓칠 수 있다. 적어도 두 번은 읽어야지 이 책이 가진 진가를 알 수 있을 것이다. 마지막으로, 이 책에서 접한 수많은 문구 중 인상 깊은 것을 몇 가지 적어보고자 한다. 극히 일부만 인용했는데, 이 책은 이런 주옥같은 문장들의 보고라고 할 수 있다. 직접 찾아보도록 하자.

"처음 하는 것은 대부분 실패하며, 이것은 아무리 용의주도하게 준비한다고 해도 마찬가지다. 따라서 실패해도 좋은 상황을 만들어서 가능한 한 빨리 실패하는 편이 문제점을 바로 알 수 있고 고치기도 쉽다."

"중요한 것은 '이런 기능이 있다'가 아니며, 기능 사용법을 반복 연습하는 것은 더더욱 아니다. 해당 기능이 목적을 이루기 위해 도움이 된다는 것을 이해하는 과정이라는 것이 중요하다. 이런 과정을 통해 자연스럽게 언어를 익힐 수 있고, '무엇을 해결하기 위한 도구인지' 그 본질을 이해하고 사용할 수 있게 된다."

2015년 9월

김완섭

베타리더 후기

김은솔(연세대학교)

프로그래밍에 관심을 두게 된 것은 전공과목에서 필요한 자료 처리 때문이었습니다. 자료전산처리에 관한 수업을 듣고 생긴 관심을 책을 보면서 어떻게든 해보려는 것은 너무 험난한 길이었습니다. 책의 첫머리에서 벗어나지 못한 지가 1년째, 그러다 이번 베타리딩을 통해 《나의 첫 프로그래밍》이라는 책을 만났습니다. 제가 지금 하는 것이 무엇인지, 앞으로 할 것은 무엇인지, 그리고 무엇을 위해 하려는지를 알려주는 이정표 같은 책이었습니다.

김정현(서울과학기술대학교)

전공이 컴퓨터공학임에도 프로그래밍과는 약간 담을 쌓은 저에게 이 책은 정말 흥미롭게 다가왔습니다. 프로그래밍하면서 어려움을 겪었던 건 언어 구조가 아니라 '어떻게 프로그램을 만들어야 하는가?'에 대한 것이었는데. 특히 이 책은 단순하게 코딩을 강조하는 것이 아닌, 생각하면서 만드는 합리적인 프로그래밍을 강조하고 있어서 무척 마음에 들었습니다. 책의 수준은 전반적으로 무난했습니다. 정말 생초보도 무조건 따라 하기만 하면 하나의 멋진 게임이 완성된다는 것은 무척이나 매력적이죠. 더군다나 무조건 따라 하는 부분과 함께 문제가 발생했을 때 어떻게 고찰해볼 것인가에 대한 저자의 철학도 담겨 있어서 읽는 데 거부감이 들지도 않았습니다. 특히 저와 같은 전산 전공자이면서도 프로그래밍을 멀리하려고 하는 이상한 사람에게도 파이팅을 불어넣어 주는 것 같아 읽는 내내 기분이 좋았네요. :)

노승헌(아카마이 테크놀로지스 코리아)

10년 이상 IT 분야에서 일하면서 많은 프로그램을 만들어 왔습니다. 중요한 업무용 프로그램도 있었고, 스크립트로 만든 간단한 유틸리티도 있었습니다. 그런데 이 책을 읽기 전까지 '왜 이렇게 프로그래밍을 하게 되는 거지?'라는 생각을 별로 해본 적이 없었더군요. 코드의 이면에 숨겨진 '왜'에 대한 고민을 저자와 함께 떠날 수 있었던 즐거운 시간이었습니다. 책을 처음 펼쳐들면서 'Sunaba'라는 생소한 환경에 한 번 놀라고, '왜 이렇게 하는 거지?' 하는 궁금증을 자아내는 '메모리에 직접 엑세스하기' 방식에 또 한 번 당황했습니다. 하지만 한 장 한 장 넘어가면서 간단한(?) 테트리스 게임을 Sunaba 환경으로 구현하면서 근대적인 언어들이 가지고 있는 방식으로 코드를 개선해나가는 과정은 정말 흥미진진했습니다. 저자의 질문들을 보고 함께 고민하면서 '아~!' 하며 무릎을 탁 치는 순간들도 여러 번 있었습니다. 이 시간 이후부터는 코드 한 줄 한 줄을 적을 때마다 '왜'에 대한 생각이 꼬리에 꼬리를 물 것만 같은 느낌입니다!

장윤주(탱그램팩토리)

프로그래밍 언어를 전혀 모르는 사람도 자체 설계한 언어(Sunaba)와 동작 환경을 이용해 간단한 게임(테트리스)을 만들면서 프로그래밍 세계에 입문할 수 있도록 해주는 책입니다.

장진주(줌인터넷)

프로그래밍 완전 초보가 읽기에도 부담스럽지 않은 내용입니다. 전문적인 프로그래밍 언어를 선택하기 전에 '프로그램'이란 게 어떤 고민을 거쳐 어떻게 만들어지는지를 직접 체험하기에 아주 좋은 책입니다. 사실, 책 초반에는 예제용 압축 파일을 푸는 것부터 시작해서 브라우저에서 다운로드한 파일이 어디에 저장되는지 등을 다뤄 너무 초보자 취급을 받는 느낌이 들었습니다. 하지만 이후로는 프로그램 설계 방법을 전반적으로 친절하게 잘 설명해줍니다. 친절이 조금 과다한 부분들도 있었지만, 초보자 입장에서는 덜 친절한 책보다는 친절한 책이 훨씬 좋습니다. :)

최아연

책을 읽으면서 프로그래밍을 처음 배우던 시절이 많이 생각났습니다. 입문자가 처음 접할 당시에는 쉽게 생각해내기 어려워서 헤맸던 개념, 생각하는 단계와 방식을 정말 쉽게 설명해주는 책인 것 같습니다. 책을 처음 보고 한글로 된 코드에 당황했지만 말이죠(ㅋㅋㅋ). 내용이 쉽게 잘 쓰여 있어 프로그래밍을 처음 배우려는, 혹은 프로그래밍을 배우려다 포기했던 분들에게 자신감을 불어넣기에 충분한 책인 것 같습니다.

한홍근(고려대학교)

첫 베타리더 활동을 통해 여러 도움을 얻었습니다. 첫째, 원고 속 저자의 충고를 통해 프로그래밍에 대해 다시 한 번 생각해보게 되었습니다. 둘째, 책을 읽으며 깊게 생각하는 방법을 익혔습니다. 셋째, 프로그래밍을 어려워하는 제 주변의 친구, 동생들에게 공부 방향에 대해 도움을 줄 수 있게 되었습니다. 베타리딩 시작 전에 때마침 고등학생을 대상으로 한 프로그래밍 캠프에서 조교로 활동하면서 어떻게 해야 프로그래밍을 쉽게, 그리고 겁먹지 않게 가르쳐 줄 수 있을까를 고민하고 있던 터였습니다. 이 책을 읽고 검토하면서 그에 대한 답을 어느 정도 얻은 것 같고, 이 책을 동생들에게 자신 있게 추천할 예정입니다.

홍성남

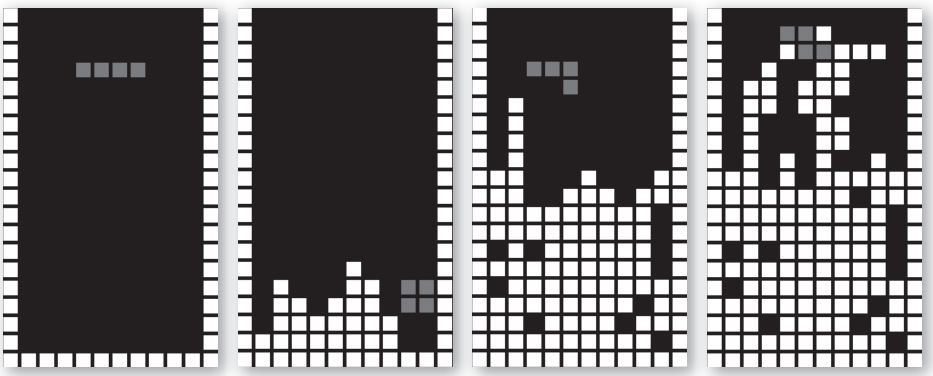
프로그래밍을 처음 배울 때 초보자에게 가장 중요한 것은 '내가 무엇을, 어떻게 할 수 있는가?'를 아는 것일 겁니다. 이 책의 저자는 게임 제작이라는 커다란 '무언가'를 제시해주고, 이어서 '어떻게'에 해당하는 사고방식과 습관을 알려준다는 점에서 이 책에 후한 점수를 주고 싶습니다. 그리고 신기하게도 이 책은 소설이 아닌데도 속도감이 있습니다. 다소 느린 속도감이긴 하지만 말입니다. 걸음마를 하며 넘어지다 보면 어느새 걷고 있는 자신을 발견할 수 있을 겁니다. 또한, 이 책은 재미가 있습니다. 보통은 초심자를 위한 책인데도 변수나 조건문, 반복문에서 시작하는 경우가 대다수인데, 이 책은 간단한 테트리스 게임을 직접 만들며 프로그래밍을 배우다 보니 어렵거나 지루하지 않게 프로그래밍을 배울 수 있을 것 같습니다. 컴퓨터 전공자나 프로그래밍 경력이 꽤 있는 분들은 어떨지 모르지만, 비전공자나 초보자에게 이만한 책은 없다고 생각됩니다.

CHAPTER

0

이 책은 누구에게 무엇을 제공하는가?

먼저 다음 그림을 보자.



이 책은 떨어지는 조각들을 맞추는 게임을 제작하면서 컴퓨터 프로그램(이하 프로그램) 개발 방법을 이해하고 자기 것으로 만들기 위한 입문서다. 그러므로 여러분이 한 번도 프로그램을 만들어본 적이 없다는 것, 그리고 컴퓨터 관련 공부를 전혀 해본 적이 없다는 것을 가정하고 설명할 것이다. 내가 원하는 것은 여러분에게 프로그램 개발 능력을 부여하는 것이다. 다른 사람이 만든 것을 수정하거나 다른 사람이 만든 것을 참고해서 만드는 능력이 아니다. 여기서 능력이란, 모든 것을 스스로 생각해서 눈에 보이는 프로그램을 만들어내는 힘을 말한다.

왜 이 책이어야 하나?

이 책의 취지는 앞에서 이야기했다. 하지만 이것만으로는 아직 이 책에 대한 흥미가 부족할 것이다. 프로그램 개발을 배우는 것이라면 이 책 말고도 다양한 방법이 있다. 학교에서 배울 수도 있고 직장에서 배울 수도 있다. 프로그램 개발 입문서는 수없이 많으며 무료 웹사이트들도 많다. 그러면 왜 이 책이어야 할까? 다른 방법과 차별화된 무언가가 있는 것일까? 이 책을 통해 배워야 하는 이유를 설명하겠다. 약간 길지만, 본론으로 들어가기 전에 꼭 짚고 넘어가도록 하자. 이 책을 선택하느냐 마느냐는 이 설명을 읽고 결정해도 늦지 않는다.

0.1 기존 학습법의 세 가지 문제

기존 프로그램 개발 관련 학습 방법에는 세 가지 큰 문제가 있다.

- 개별 요소를 따로따로 설명하지만, 이들을 조합하는 방법은 알려주지 않는다
- 전문가용 도구들을 갑자기 사용해서 시작하자마자 좌절한다
- 내용을 이해할 수 없는 사람이 있더라도 어쩔 수 없다고 생각한다

각 문제에 대해 조금 더 자세히 이야기해보겠다.

0.1.1 요소가 제각각

프로그램 개발에는 많은 도구가 필요하다. '프로그래밍 언어'라는 도구가 특히 중요하며, 개발 환경인 소프트웨어도 필요하고, 수학이나 정보 공학 같은 학술적인 지식도 필요하다. 이를 위해 각 분야마다 책이나 웹사이트가 존재하며, 수업이나 연수 등을 통해 그 정보를 얻기도 한다. 하지만 대개 그 도구들을 사용해서 어떻게 프로그램을 만들어야 하는지는 가르쳐주지 않는다.

아직 여러분은 프로그램 개발에 대해 잘 모를 테니 요리를 예로 들도록 하겠다. 냄비나 칼 사용법, 그리고 식재료에 대한 책은 많지만, 정작 요리법(레시피)이 없다고 생각해보자. 잘게 썰기나 깍둑썰기, 냄비의 종류나 사용법, 조미료 종류 등이 망라된 책이 있음에도 카레에 넣을 당근을 어떻게 썰어야 하는지 모른다. 애당초 카레에 당근을 넣어야 하는지조차 모를 수 있다.

물론, 재능과 근성이 있다면 요리법을 몰라도 어떻게든 음식을 완성할 수 있다. 먹어본 카레를 기억하면서 사용할 재료를 결정하고, 써는 방법이나 삶는 방법도 시행착오를 통해 알아간

다. 이런 경험을 통해 '삶을 때는 이런 썰기 방법을 쓴다', '이 야채에는 이 조미료가 맞다'와 같은 사실을 알 수 있다. 언젠가는 요리에 자신이 붙게 되고, 나아가 새로운 요리를 만들어낼 수도 있게 된다.

하지만 이 수준까지 도달할 수 있는 사람이 과연 몇 명이나 될까? 요리법이 있다면 당근을 넣을지 말지를, 어떻게 썰어야 할지를 고민할 필요도 없다. 다양한 요리를 요리법을 따라 만들어나가면 앞서 언급한 시행착오의 시간을 거치지 않고 요리의 기본을 익힐 수 있다. 재능이나 근성은 그다지 중요하지 않다.

요리 초보자가 과연 '조림 이론'이나 '조미료 백과사전' 같은 책을 살까? 하지만 프로그램 개발 책은 대개 그런 식이다.

프로그램에 요리법 책이 있을까?

프로그램에 요리법 책이 없는 데는 이유가 있다. 먼저 프로그램 종류에 따라 사용하는 기술 차이가 크다. 워드프로세서와 게임에서 사용하는 기술이 다르다. 요리에서도 초밥과 카레를 만드는 방법은 많이 다르며, 초밥 책을 보면서 카레를 만들 수는 없다. 또한, 프로그램을 한 권의 책으로 다 설명하기에는 그 범위가 너무 방대하다. 예전에 내가 개발했던 어떤 게임은 7명의 프로그래머가 1년 동안 매일 10시간 넘게 작업해서 완성했다.[1] 이렇게 방대한 내용을 한 권의 책으로 설명하는 것은 무리다.

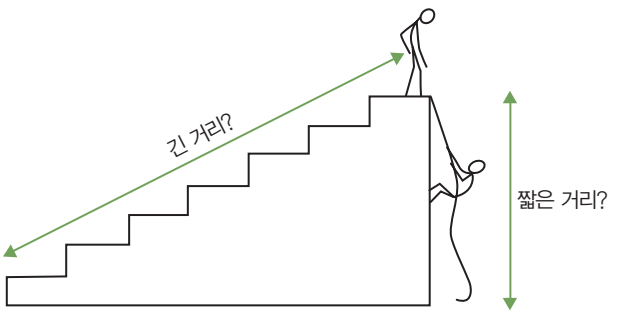
프로그램 개발 책이 냄비나 칼 사용법처럼 개발 요소에 치중하고 있는 것은 이런 이유 때문이다. 물론, 전체를 보여주고자 하는 고민이나 노력은 반영되어 있다. 하지만 전체를 보여주기 위해서 세부 내용은 생략해버린다. '냄비 세트를 사서 지시하는 대로 야채를 썰고 조리기만 하면 돼'라고 말하는 것과 같다. 이런 내용으로는 제대로 배울 수 없다.

0.1.2 전문가용 도구가 초래하는 좌절

많은 접시를 한 번에 닦는 식기세척기는 가게를 효율적으로 운영하기 위해 필요한 도구다. 하지만 지금 당장 사용하지도 않는데 전문가가 되면 필요하다는 이유로 전문가용 도구를 배울 이유가 있을까? 오히려 배워야 할 다른 중요한 것들이 많다. 그런데 프로그램에 관해서는 이 당연한 의문이 무시되고 있다. 처음부터 복잡한 도구를 사용해서 수십 명 규모로 개발할 때나 필요한 개념과 사용법을 무조건 주입시킨다. 그러니 좌절하는 것도 당연하다.

1 이 정도 규모는 그다지 큰 규모의 게임이 아니다. 프로그래머가 50명 넘게 참여하는 게임도 적지 않다.

전문가와 같은 수준의 도구를 사용한다고 하면 오히려 동기부여가 되는 사람도 있긴 하다. 이들은 교육용이라는 얘기를 들으면 싱거워한다. 초보자들은 야심에 불타오르기 쉬워서 전문가가 사용하는 도구를 동경하며 그것이 가장 빠른 지름길이라고 생각하기 쉽다. 하지만 그것은 "한 계단씩 오르면 어느 세월에 올라가냐? 나는 로프를 타고 올라갈 거야!"라고 하는 것과 같다. 물론, 그렇게 올라갈 수 있는 사람도 있다. 훌륭한 사람이다. 하지만 나처럼 평범한 사람에게는 계단이 필요하다.



0.1.3 이해하지 못해도 어쩔 수 없다

마지막 한 가지는 더 심각한 문제다. 따라올 수 없는 사람이 있어도 어쩔 수 없다고 생각하는 것이다. 프로그래밍 경력을 쌓아서 강사가 되거나 책을 쓰는 사람은 당연히 프로그래밍을 잘 아는 사람이다. 잘 아는 사람은 그렇지 못한 사람에 대해 무지하다. 배우는 사람이 따라올 수 없는 것은 본인의 노력이 부족하기 때문이라고 치부해버린다.

배우는 입장에서도 책임은 있다. 대부분 노력이 부족하고 재능의 차이도 크다. 하지만 그렇다고 해서 배우는 사람이 잘못된 것은 아니다. 가르치는 입장에서 최선을 다해야 한다. 내가 느낀 바로는 가르치는 사람이 최선을 다하는 경우는 그리 많지 않았다.

또한, 얼핏 프로그래밍에 적성이 없는 것처럼 보인다고 해서 정말로 적성이 없는 것은 아니다. 본인이 적성에 안 맞는다고 말한다고 해도 그것이 정말로 적성에 안 맞는 것을 의미하는 것은 아니다. 나는 이 사실을 대학생이나 후배들을 가르치면서 깨달았다.

사람은 서로 다른 인생을 살아왔으니 좌절하는 방법도 제각각이다. '겨우 이 정도의 일'이라고 생각되는 것에 좌절하는 사람도 있다. 하지만 그 단계를 넘으면 계속 성장해나갈 수 있다. 오히려 초기에 다른 사람이 좌절하지 않는 자잘한 부분에서 제대로 실패를 경험한 사람이 나중에 더 빠르게 성장하는 경우도 있다.

'마이너스 곱하기 마이너스가 왜 플러스가 되는지 이해할 수 없다'고 생각하는 학생이 오히려 수학에 적성이 있을 수도 있다. 이것을 '재능이 없어', '서툴러'라고 단정 지으면 안 된다. 나는 이런 사람들에게 도움을 주고 싶다.

0.2 이 책은 무엇이 다른가?

이 책은 적어도 나의 관점에서는 이런 결점들을 없애고자 많은 노력을 했다. 기존 프로그램 개발 책과 다른 점은 다음과 같다.

- 하나의 프로그램을 처음부터 끝까지 만들어본다
- 전용 도구를 제공한다
- 정말로 처음인 사람을 대상으로 한다

0.2.1 하나의 프로그램을 처음부터 끝까지 만들어본다

이 책을 요리책에 비유하자면, '카레를 통해 요리를 배우는 책'이라고 할 수 있다. 어디에 사용되는지 모르는 요소들을 따로따로 설명하는 일은 없다. 모든 요소를 카레를 만들기 위한 수단으로 소개한다. 그리고 소개하는 수단에 대해서는 그것을 어디에 사용하고 왜 필요한지를 설명한다. 이런 과정을 이해하지 않고 공부하고자 하는 사람에겐 이 책이 필요 없을 것이다.

이를 위해 포기해야 하는 것

물론, 이런 설명 방식에 장점만 있는 것은 아니다. 여러분이 카레에 관심이 없다면 이 작전이 맞지 않을 수도 있다. 이 책 초반에 소개하는 게임 화면에 매력을 느끼지 못한다면 이 책이 매우 지루할 수도 있다. '카레라면 모두가 좋아하겠지'라고 생각하지만, 여러분이 어떤 취향을 가졌는지 알 방법이 없다.

또한, 카레에 필요한 것만 설명하고 있기에 뭔가 부족하다고 느낄 수도 있다. 잘게 써는 방법이나 프라이팬 같은 도구를 사용하지 않고 간장도 사용하지 않는다. 즉, 하나의 게임을 만들면서 모든 기술을 포함하기는 어렵다.

순서 문제도 있다. 카레를 만드는 과정을 순서대로 설명하므로 분류나 정리 등은 일절 하지 않았다. 1장 써는 법, 2장 조미료, 3장 조리기처럼 분류해서 설명하는 것은 무리가 있다. '양파를

다져서 볶은 후 당근을 깍둑썰기한다'고 설명하면, '다지기'와 '깍둑썰기'에 대해 별도로 설명해야 한다. 하지만 이런 방식은 개별 기술을 익히기 위한 효율적인 방법이 되지 못한다.

새로운 도구를 사용하기 전에 그 도구가 왜 필요한지 이해시키는 방법에도 문제가 있다. 왜 필요한지를 이해하려면 그 도구가 없을 때 어떤 문제가 있는지를 알아야 한다. 하지만 이것은 말로 설명한들 이해하기 어렵다. 직접 체험하는 수밖에 없다. 이 책에서는 여러분이 충분한 시행착오를 거치기 전까지는 도구를 내놓지 않을 것이다. 따라서 책을 읽는 데 시간이 꽤 걸릴 수도 있다. 하지만 나는 이런 과정이 꼭 필요하다고 생각한다.

0.2.2 전용 도구를 제공한다

전문가가 될 수 있는 기술을 익혔다면 최고의 도구는 전문적인 도구가 될 것이다. 하지만 초보자인 여러분에게는 아직 최고의 도구란 없다. 이는 당연한 이야기이지만, 신기하게도 대부분의 사람은 인정하지 않는다. 최고가 되기 위해서는 최고의 전문가에게서 배우는 것이 가장 빠른 길이라는 얘기를 가끔 듣지만, 노벨상을 받은 물리학자가 초등학교에서 가르친들 무슨 소용인가!

그런 의미에서 이 책에서는 전용 도구를 제공한다. 프로그래밍 언어, 개발 환경, 그리고 그에 관련된 몇 가지 용어나 개념까지 초보자인 여러분을 위해 준비했다. 최소한의 기능만 가지고 있으며, 배워야 할 규칙도 최소한으로 만들었다. 이 책 말고 다른 곳에서 별도의 지식을 찾아 헤맬 필요도 없다.

이를 위해 포기해야 하는 것

물론, 관점에 따라 문제가 될 수도 있다. 먼저 이 책만으로는 전문가가 될 수 없다. 전문 도구는 별도로 배울 필요가 있다. 나는 이것이 멀리 돌아가는 길이라고 생각하진 않지만, 여러분이 그렇게 생각한다면 방법이 없다.

또한, 여기서 제공하는 프로그래밍 언어로는 대단한 것을 만들수는 없다. 판매용으로 만들어진 것처럼 규모가 크고 복잡한 것은 만들 수 없다. 개발 환경도 빈약해서 어느 정도 익숙해지면 기능 부족 때문에 불만이 생길 수도 있다.

다른 한 가지는 습관의 문제다. '처음부터 제대로 된 방법을 배우지 않으면 나쁜 버릇이 생긴다'고 위협하는 사람이 많다. 처음부터 전문 도구를 사용해서 전문가가 하는 방법을 배워야 좋은 습관이 든다는 의견이다. 동의하지는 않지만, 그렇게 생각하는 사람이 많다는 것을 알고 있다.

그리고 전용 언어를 사용한다는 것은, 관련 정보가 이 책밖에 없다는 것을 의미하기도 한다. 이 책을 통해 얻을 수 있는 정보만을 가지고 고민하고 조합해야 한다. '다른 것을 조사할 필요가 없다'는 말은 듣기 좋지만, '다른 것을 조사할 수 없다'는 말은 장점으로 들리지 않는다. 따로 조사해야 하는 기술은 쉽게 자기 것이 되기 어려울 뿐만 아니라 다른 사람의 방법을 이해하는 데에도 도움이 되지 못한다. 따라서 이런 방식의 학습은 완전히 배제하고 있다. 지금 현재의 여러분에게는 오히려 외부 기술을 배제하는 것이 도움된다고 판단했기 때문이다.

0.2.3 정말로 처음인 사람을 대상으로 한다

'초보자 대상'이라는 표현은 애매하므로 사용하지 않겠다. '프로그램을 처음으로 만들어보는 사람'이라고 대상을 분명히 해두고 싶다. 그래서 가능한 한 정중하면서도 차근차근 설명하였으며, 일상의 예를 많이 제시해서 쉽게 이해할 수 있도록 했다. 또한, 용어 하나도 여러 번 검토를 거쳐 쉽게 이해할 수 있는 용어를 선별했다. 추상적인 개념은 될 수 있으면 뒤로 미루고 구체적인 예를 먼저 설명하도록 했다. 결과적으로 이 책은 다음과 같은 특징을 가진다.

다른 책에 의존하지 않는다

책을 선택할 때도 노력이 필요하다. 책을 사는 것도 고생이고 돈도 든다. 또한, 책을 읽는 것도 쉽지 않다. 여러 권의 책이 필요하다는 것은 이렇게 귀찮고 돈까지 드는 일을 몇 번이고 반복해야 함을 의미한다. 배움에 대한 초보자의 의욕을 꺾기에 충분하다. 하지만 이 책은 다른 책의 도움 없이도 게임을 완성할 수 있도록 구성했다. '프로그래밍 언어의 문법은 다른 책을 참고', '개발 환경 사용법은 또 다른 책을 참고'와 같은 식의 내용은 없다.

전제 조건을 가능한 한 최소화했다

이 책은 여러분이 처음으로 프로그램 개발을 경험하고 있다고 가정하고 있다. 변수, 함수, 루프, 분기 등의 용어를 전혀 모른다고 가정하고 있다.[2] 프로그램 만드는 순서를 전혀 몰라도 된다.

영어도 필요 없다. 프로그래밍이라고 하면 영어권에서 만들어진 것이므로 영어가 필요하다고 생각할 수 있지만, 그것은 전문가가 된 이후의 얘기다. 지금은 필요 없다.[3] 수학도 필요 없다. 초등학교에서 배우는 산수 정도로 충분하다.

---

2 이 용어들 자체가 등장하지 않는다.

3 영어식 프로그래밍이 필요해지면 그때 가서 배우면 된다. 나중에 필요하니 지금 배워둬야 한다고 말하는 사람을 나는 믿지 않는다. 필요해지기 전까지는 최선을 다해 배울 이유가 없다.

하지만 아쉽게도 윈도우의 기본적인 사용법은 알고 있어야 한다. 프로그램 실행 방법이라든가 드래그 앤드 드롭(drag & drop), 더블 클릭 같은 조작 방법을 알고 있다고 가정하며, '파일'이 '폴더'에 저장되어 있다는 것 정도는 이해하고 있어야 한다. 물론, 윈도우가 설치된 컴퓨터도 필요하다.[4]

이를 위해 포기해야 하는 것

물론, 이런 특징은 단점이 되기도 한다. 책을 읽다 보면 설명이 너무 '장황하다'고 느낄 수도 있다. '몇 번이고 같은 설명을 반복해서 할 필요는 없어', '그건 알고 있으니 다음 내용을 이야기해줘', '더 좋은 방법이 있는데 왜 이런 불편한 방법을 사용하는 거야'와 같은 불만이 있을 수도 있다.

그리고 대단한 무언가를 만들지는 못한다. 그저 이 책에서 다루는 '게임'을 만들 수 있을 정도다. 이미 말했듯이 규모가 큰 프로그램을 만드는 방법을 다루려고 하면 세부 내용을 제외할 수밖에 없다. 한 권의 책에 모든 것을 담을 수 없기 때문이다. 세부 내용도 버리지 않기 위해서는 프로그램 규모를 줄여야만 한다. 모두가 쉽게 오를 수 있는 언덕을 만들려면 언덕의 높이를 낮게 잡는 수밖에 없다.

이런 결점을 인정하고 '더 높은 곳에 오를 수 있도록 100m의 경사 길을 걸어갈 수 있을 정도'의 책을 만들려고 노력했다.

0.3 그러면 시작해보자!

정리하면 다음과 같다.

- 하나의 프로그램을 완성하는 과정을 통해 프로그램 개발 방법을 배운다
- 사람이 이해하기 쉬운 전용 프로그래밍 언어를 사용한다
- 프로그램을 처음으로 만들어보는 사람을 대상으로 하며, 이 한 권을 통해 기본적인 기술을 모두 익힐 수 있다

만약 이 책이 자신에게 맞지 않는다고 느꼈다면 아까운 돈과 시간을 할애하지 않아도 좋다. 하지만 여기까지 읽고서 내가 말하고자 한 것을 이해했다면, 나도 여러분을 위해 최선을 다할 것이다.

그럼, 시작해보자!

---

4 윈도우 7과 8을 주요 대상으로 하며, Vista와 XP에서도 동작한다. 2000에서도 동작할 수도 있다. 기타 iOS 등은 요청이 있으면 개발할 생각이다.

CHAPTER

1

프로그램을 만들기 전에

무엇부터 시작해볼까? 물론, 이것은 책이므로 이미 내 머릿속에는 시작할 내용이 정해져 있다. 결론부터 말하자면, 여러분이 가장 먼저 해야 할 것은 전용 프로그래밍 언어로 프로그램을 만들 수 있도록 필요한 소프트웨어를 준비하는 것이다. 그다음에는 전용 프로그래밍 언어를 이용해서 화면에 사각형을 그릴 것이다. 이것이 여러분의 첫 프로그램이 될 것이다.

하지만 이런 내용은 내가 정한 것이다. 가장 적합한 순서라고 생각했으므로 그렇게 구성했지만, 어디까지나 나의 생각이다. 그래서 이번 장에서는 '게임을 만들려면 어떻게 하면 좋을까?'라는 의문에서부터 시작해서, 구체적으로 어떤 작업을 해야 하는지까지 설명하도록 하겠다. 이런 설명을 통해 왜 사각형을 그리는 것부터 시작하는지 이해할 수 있고, 어떤 식으로 프로그램 개발을 공부하면 좋은지 알 수 있을 것이다.

1.1 게임을 만들려면 어떻게 해야 할까?

첫 번째 질문부터 시작해보자. 이 책에서 다루는 게임은 어떻게 만들면 좋을까? 대답은 정해져 있다. 프로그램 개발 방법을 공부하면 된다. 이 책도 그런 목적으로 있는 것이다. 하지만 이 대답이 틀린 것은 아니지만, 그다지 의미는 없다.

카레 만들기를 생각해보면 바로 알 수 있다. '카레를 만들기 위해서는 요리를 배우면 된다'고 말하는 것은 의미가 없다. 지금은 요리를 배우기 위해 카레를 만드는 것이다.

그러면 의미 있는 대답은 무엇일까? 카레로 말하자면, '카레를 만들기 위해서는 먼저 무엇을 하면 좋은지'를 알면 된다. 즉, 의미 있는 대답은 '먼저 이것을 한다'를 정할 수 있는 대답이다. 카레 만들기에서 제일 먼저 해야 할 것은 양파를 볶는 것이다. 그렇다면 먼저 배워야 할 것은 '다지기' 방법이다.

이것을 게임에 적용해보자. 게임을 만들기 위해서는 먼저 무엇을 해야 할지 정해야 한다. 이를 위해서는 게임에 필요한 것이 무엇인지 생각할 필요가 있다. 그리고 프로그램 개발 과정을 통해서 게임을 구현하므로 '게임을 만드는 데 필요한 프로그램 개발 기술이 무엇인지'를 생각하면 된다.[1]

1.1.1 게임에 필요한 프로그램 기술은 무엇인가?

그런데 '프로그램 기술'이 무엇인지를 아직 모르는 상황이다. '프로그램이 무엇인지'조차 모를 수도 있다. 또한, 여기서 말하는 '게임'이 어떤 게임인지도 잘 모른다.

요리에 비유하자면, '요리 기술'이 무엇인지도 모른 채 '요리가 뭐야?'라고 묻는 것과 같다. 혹은 카레 자체를 먹어본 적이 없을 수도 있다. 또한, 카레를 먹어본 적이 있더라도 냄비나 식칼 같은 도구를 본 적이 없어서 어떻게 만드는지 전혀 감이 오지 않을 수도 있다. 여러분은 아직 프로그램에 있어 무엇이 식칼이고 무엇이 냄비인지도 모르는 상태다.

따라서 '프로그램이란 무엇인가'를 알아야 한다. 하지만 그 전에 먼저 카레를 먹어보도록 하자. 먹어본 적이 있더라도 자신이 생각한 것과 다를 수도 있다.

1.1.2 이 책에서 말하는 게임이란?

카레를 먹어보지 않고 카레를 만들 수 없듯이, 먼저 게임을 가지고 놀아보자.[2]

이 책에서 다루는 게임은 테트리스이지만, 아쉽게도 내가 준비한 것은 오리지널 테트리스 게임과는 많이 다르다. 기능을 상당히 단순화한 것이다. 말보다는 게임을 직접 실행해보는 것이 이

1 프로그램을 만들지 않고 게임을 만드는 방법도 있다. 프로그래머를 찾아서 시키면 된다. 게임을 만드는 것이 목적이고 프로그램 개발은 수단에 불과하다면, 이 책을 읽는 것이 정말 필요한지부터 다시 생각해볼 필요가 있다.

2 먹어본 적이 없는 것을 만드는 것이 불가능한 것은 아니다. 하지만 만든 것이 정말 카레인지 모르는 상태가 된다. 슬픈 일이지만 실제로 있는 일이다.

해가 빠를 것이다. 이를 통해 무엇을 만들어야 하는지도 명확해진다. 또한, 이 게임의 설치 과정을 통해서 책에서 사용할 프로그램 실행 환경도 함께 준비하도록 한다. 단순한 준비 과정이므로 빨리 끝내도록 하자.

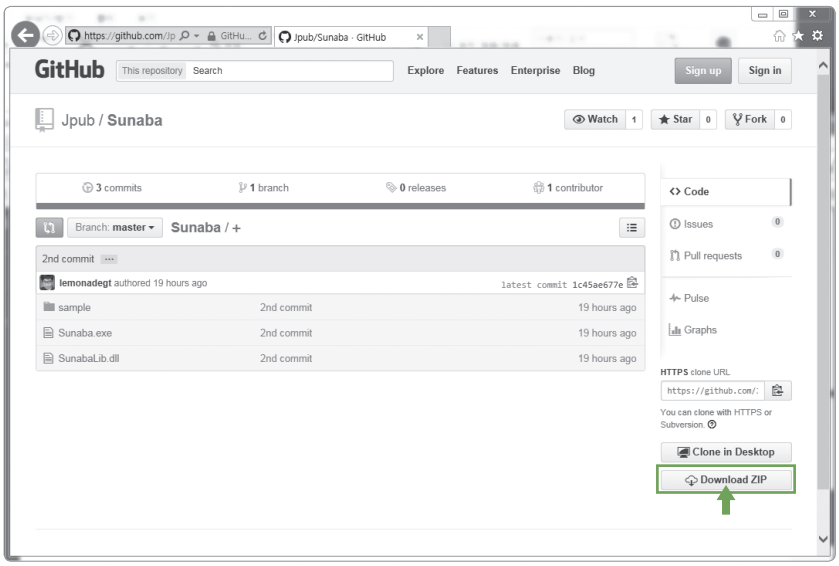
예제 게임 실행 방법

프로그램을 만들려면 '프로그래밍 언어'라는 것이 필요하다고 앞에서 설명했다. '게임'도 프로그래밍 언어로 작성돼 있다.

이 책에서 사용하는 언어의 이름은 Sunaba(砂場)다. 이름의 의미는 '모래밭'이다. 실용적인 것은 만들 수 없는 장소이지만, 뒹굴어도 다치지 않는 곳이다. 모래밖에 없으므로 상상력이 없으면 아무것도 할 수 없다. 어릴 때만 좋아하지, 크면 잊혀지는 장소다. 그런 모래밭에 대한 생각이 프로그래밍 언어에 그대로 반영돼 있다.

어떠한 사정 때문에 Sunaba로 작성된 프로그램은 전용 소프트웨어에서만 동작한다. 여기서는 이 전용 소프트웨어를 설치해보겠다. 이 소프트웨어는 인터넷에서 다운로드해서 설치하면 된다.[3] 웹 브라우저를 통해 다음 주소에 접속한 후 다운로드한다.

URL *https://github.com/Jpub/Sunaba*

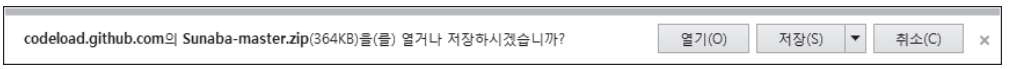


3 광디스크(CD-ROM, DVD 등)를 부록으로 제공하는 것을 고려했지만 책 가격이 너무 높아진다. 그리고 지금은 광디스크 장치를 가지고 있지 않는 사람도 많다.

화면의 오른쪽 하단에 있는 'Download ZIP' 버튼을 클릭하면 프로그램이 다운로드된다. 다운로드한 파일은 zip 형식의 파일로 압축을 풀면 내용물을 확인할 수 있다. 'zip'이나 '압축 풀기'라는 용어를 모르는 사람이 있을 수 있으니 조금 더 자세하게 설명하도록 한다. 여기서는 윈도우 7을 사용해서 설명한다. 압축 해제 방법을 알고 있다면 건너뛰어도 좋다.

다운로드와 압축 풀기

윈도우 7과 IE(인터넷 익스플로러)를 사용하고 있다면 다운로드 시에 다음과 같은 화면을 볼 수 있다.



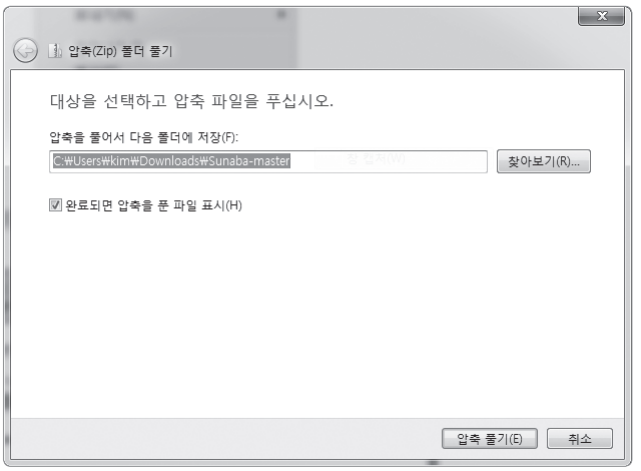
여기서 '저장'을 누른다. 다운로드할 위치를 지정하고 싶으면 '저장' 버튼 옆의 화살표를 클릭한 후 '다른 이름으로 저장'을 선택하면 된다. 선호하는 위치가 없다면 아무것도 건드리지 말고 바로 저장 버튼을 누르면 '다운로드' 폴더에 저장된다. 폴더 위치를 모를 경우 다운로드가 끝나면 나오는 화면에서 '폴더 열기'를 클릭하면 된다.

역주 이 내용은 IE 버전 10 이상의 내용으로, IE 9 이하에서는 다운로드 방식이 다를 수 있다.

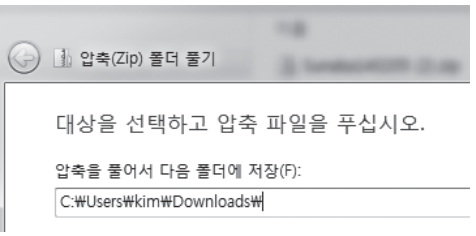
다운로드가 끝난 다음 Sunaba-master.zip 파일을 오른쪽 클릭하면 다음과 같은 메뉴가 뜬다. 여기서 '압축 풀기'를 선택한다.

역주 윈도우 XP 이하 버전에서는 압축용 소프트웨어를 별도로 설치해야지 '압축 풀기' 메뉴를 볼 수 있다.

그러면 다음과 같이 압축을 어디에 풀지를 지정하는 화면이 뜬다.



특별히 선호하는 위치가 없다면 그대로 오른쪽 하단의 '압축 풀기' 버튼을 클릭하면 된다. 하지만 다음 화면처럼 마지막 부분의 파일명을 지우고 'Downloads'만 남겨두고 '압축 풀기' 버튼을 누르면 이후 작업이 더 수월해진다. 다운로드 폴더에 Sunaba-master라는 폴더가 생성되고 필요한 것이 모두 담겨 있는 것을 확인할 수 있다.[4] 이후 실습의 편의를 위해 Sunaba-master 폴더를 바탕 화면으로 옮기도록 한다. 이 책에서도 바탕 화면에 해당 폴더가 있다고 가정하고 설명을 진행한다.

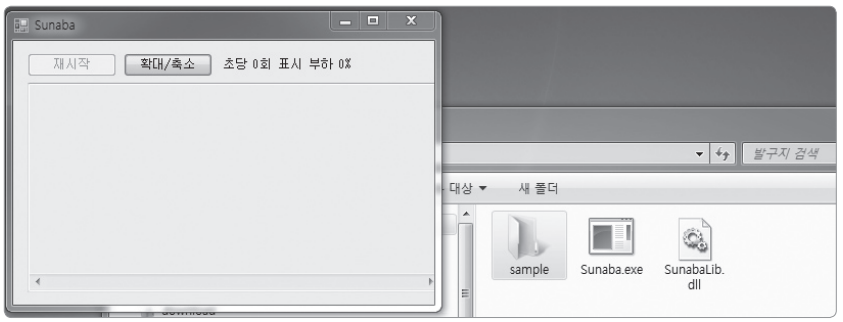


폴더를 열어서 프로그램을 실행한다. 이 폴더는 다음과 같이 구성돼 있다.

'Sunaba'라는 파일[5]을 더블 클릭한다. Sunaba 화면이 뜰 것이다.

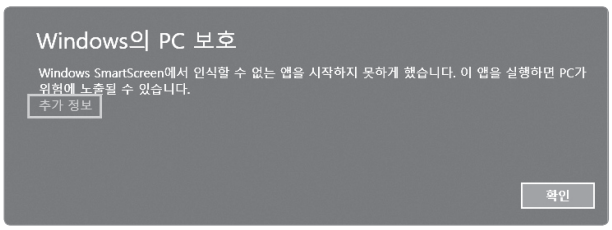
4 참고로, 이 프로그램을 더 이상 사용하지 않는 경우에는 그대로 휴지통에 버리면 된다. 이른바 언인스톨(프로그램 제거) 과정이 필요 없다.

5 윈도우 설정에 따라서는 'Sunaba.exe' 형식으로 '.exe'가 함께 표시될 수도 있다. 이 경우는 윈도우를 어느 정도 알고 있는 사람으로, 옵션을 고쳐서 확장자를 표시한 것이다.

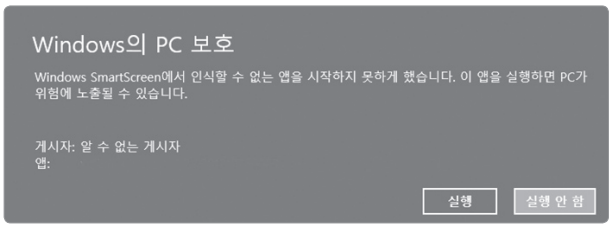


윈도우 8인 경우

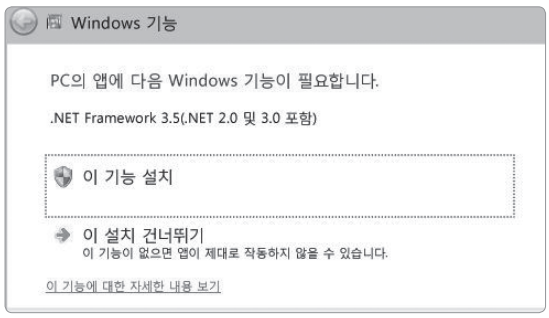
단, 윈도우 8에서는 다음과 같은 화면이 뜨면서 실행이 되지 않을 수도 있다.



이때는 '추가 정보'를 클릭하면 다음과 같은 화면을 볼 수 있다.



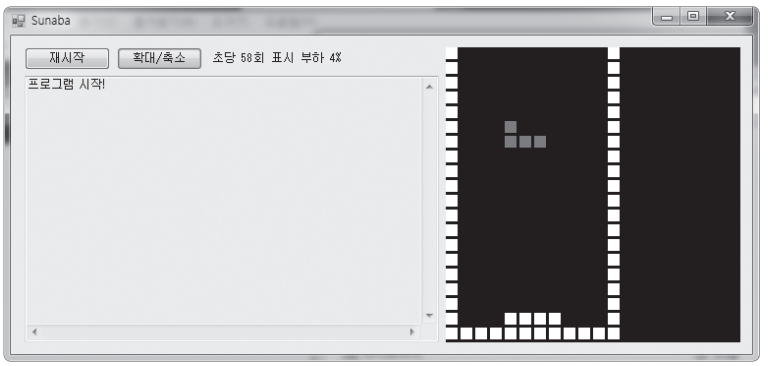
여기서 '실행'을 클릭하면 된다. 또한, 다음 화면이 뜰 수도 있다.



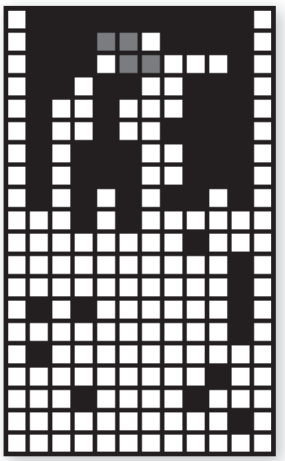
이때는 '이 기능 설치'를 선택한다. 설치에 조금 시간이 걸리니 기다리도록 하자. 설치가 끝나면 Sunaba가 실행될 것이다.

샘플을 실행해보자

다음은 Sunaba 폴더 내에 있는 'sample'이라는 폴더를 연다. '게임'[6]이라는 파일이 있으니 이것을 Sunaba 창에 드래그 앤드 드롭한다. 그러면 게임이 동작하기 시작할 것이다. 키보드의 좌우 버튼을 눌러서 떨어지는 빨간 벽돌을 움직일 수 있고, 위쪽 화살표 키를 누르면 벽돌이 회전한다.



위쪽 화살표 키로 회전하거나 좌우 키로 움직여가면서 빈틈없이 열을 채우면 해당 열이 사라지고 위에 있는 열들이 아래로 내려온다. 블록이 계속 쌓여서 꼭대기까지 쌓이면 더 이상 블록 조작이 되지 않는다.



6 이름 뒤에 '.txt'가 숨겨져 있다.

이 상태에서 게임을 재시작하고 싶다면 '게임' 파일을 다시 드래그 앤드 드롭하면 된다.[7]

판매되고 있는 실제 테트리스에서는 블록이 떨어지는 속도가 빨라지는 등 다양한 추가 규칙이 존재하지만, 이 책에서 사용하는 게임은 어디까지나 샘플이므로 그런 기능은 없다.[8]

이것이 여러분이 만들어야 할 게임 프로그램이다. 잠시 게임을 즐기면서 어떻게 하면 이런 게임을 만들 수 있는지 생각해보자. 참고로, Sunaba 창에 있는 '확대/축소' 버튼을 누를 때마다 화면 크기가 바뀐다. 화면을 크게 보고 싶거나 작게 하고 싶을 때 누르면 된다.

1.1.3 프로그램이란 무엇인가?

내가 만든 게임을 실행해보았다. 카레로 말하자면, 먹어봤다고 할 수 있다. 당근이나 고기가 들어 있다는 것도 알았고, 어떤 맛인지도 알았다. 하지만 '이것을 만들려면 어떻게 하면 좋을지'에 대해서는 아직 정보가 부족하다. 주방을 들여다볼 필요가 있는 것이다.

하지만 프로그램을 만드는 사람은 요리하는 사람만큼 많지가 않다. 운이 좋아서 주변에 프로그래머가 있다고 해도 요리와 달리 보기만 해서는 알기가 어렵다. 도구가 컴퓨터밖에 없어서 잘 보지 않으면 인터넷에서 재미있는 글을 읽는 것과 별반 다르지가 않다.

프로그램은 컴퓨터 안에 존재하므로 형태가 없다. 게다가 일상에서 늘 접하는 문장이나 그림, 음악과 달리 친숙하지 않은 존재다. '프로그램을 만든다'에서 '만든다'라는 의미조차 잘 모르는 경우가 많다. 그런 의미에서 요리나 스포츠처럼 '우선은 그것을 시작한 사람을 관찰하는 것부터 한다'는 법칙이 적용되지 않는다.

그러면 '프로그램이 무엇인지' 알기 위해서는 무엇을 먼저 해야 할까?

'게임'이라는 파일의 정체

앞서 '게임'을 실행할 때 '게임'이라는 파일을 드래그 앤드 드롭했었다. 이 파일의 정체는 뭘까? 먼저 이것부터 파악해보자.

이 '게임'이라는 파일은 내가 만든 것이다. 음악이나 그림, 문장도 파일이라는 형식으로 만들어진다. 프로그램도 마찬가지다. 따라서 게임 프로그램을 만든다는 것은 게임 프로그램의 파일을 만드는 것이다. 그 파일 내부를 보면 무언가 단서를 발견할 수 있을 것이다.

---

7 '재시작'이라는 버튼을 눌러도 된다.

8 '게임고급버전' 파일을 드래그 앤드 드롭하면 좀 더 화려한 게임을 볼 수 있다. 이 책을 마지막까지 읽어서 모두 이해한 후에 어느 정도 노력을 기울이면 이 정도의 게임은 만들 수 있다.

파일 내용을 보려면 파일을 더블 클릭해보는 것이 일반적인 방법이다. 워드프로세서 파일이라면 문장이나 그림, 표 등이 보일 것이다. Sunaba용 파일에서는 무엇이 보일까? 더블 클릭하면 메모장이라는 소프트웨어가 실행돼서[9] 무언가 해괴한 문장 비슷한 것이 등장한다. 그중 일부를 추출해서 보면 다음과 같다.

```
사각형(세로, 가로, 색) 정의
    세로횟수 -> 0
    세로횟수 < 4 동안
        가로횟수 -> 0
        가로횟수 < 4 동안
            메모리[60000 + (세로 * 500) + (가로 * 5) +
                가로횟수 + (세로횟수 * 100)] -> 색
            가로횟수 -> 가로횟수 + 1
        세로횟수 -> 세로횟수 + 1
```

실은 이 해괴한 문장이 프로그램이다. 한글 단어가 많지만 '->'와 같은 기호도 있으며, 완전한 문장 형태는 아니다. 수학식 같은 것도 섞여 있다. 각 행에 있는 여백의 길이도 다르다. 참 해괴한 문장이다.

하지만 이것이 프로그램으로, '게임 프로그램을 만든다'라는 것은 이 해괴한 문장을 작성하는 것을 말한다. 이런 문장을 작성해서 파일에 저장하면 '게임을 만든다'라는 행위가 되는 것이다. 그러면 이 이상한 문장의 정체는 무엇일까? 어떤 규칙으로 작성되고 무엇을 의미하는 걸까?

1.1.4 이 해괴한 문장의 정체는?

이 문장이 프로그램이라면 어떤 규칙으로 작성된 것인지 파악해야 한다. 이 규칙을 '프로그래밍 언어'라고 한다.

프로그램은 프로그래밍 언어의 규칙을 따라 문장을 작성해서 만드는 것이다. Sunaba라는 프로그래밍 언어의 규칙을 따라 만든 문장은 Sunaba의 프로그램이 된다. 마찬가지로, C라는 프로그래밍 언어의 규칙을 따라 만든 문장은 C 프로그램이 된다. 앞서 봤던 프로그램은 Sunaba 프로그램으로, 다른 프로그래밍 언어와는 전혀 다른 외형을 가지고 있다.[10] 이 책에서는 Sunaba를 사용하므로 이후 수십 시간을 들여서 Sunaba의 규칙을 익히게 된다.

그렇다면 다음 질문은 '게임을 만들기 위해서 어떻게 Sunaba의 규칙을 배울까?'가 된다.

9 메모장 이외의 프로그램이 실행되는 사람은 이미 어느 정도 아는 사람이므로 설명을 생략한다.

10 예를 들어, 대부분의 프로그래밍 언어에서는 한글을 사용하지 않는다. '->'나 'x' 같은 기호도 다르다.

1.1.5 여러분의 목표는 어느 정도의 수준인가?

여기서 잠시 옆길로 빠지겠다. 프로그램 양에 대해서 다루고 넘어가도록 하자. 여러분은 어느 정도 양의 '해괴한 문장'을 작성해야 하는 것일까? 그것을 알면 목표까지의 거리가 어느 정도인지 감을 잡을 수 있어서 조금이나마 마음이 편해질 것이다.

앞서 본 '게임' 파일을 열어서 가장 아랫부분까지 보도록 하자.

```
패턴을미리만들어둔다()
메모리[55001] -> 1 #속도 저하 끄기
벽과바닥그리기()
세로 -> 1
가로 -> 5
회전 -> 0
종류 -> 0
이전왼쪽 -> 메모리[50006]
이전오른쪽 -> 메모리[50007]
이전위 -> 메모리[50004]
낙하카운트 -> 0
1 동안 #무한
    지금왼쪽 -> 메모리[50006]
    지금오른쪽 -> 메모리[50007]
    지금위 -> 메모리[50004]
    #이동량
    가로이동량 -> 0
    세로이동량 -> 0
    회전량 -> 0
    (지금왼쪽 = 1) * (이전왼쪽 = 0) 이면
        가로이동량 -> -1
    (지금오른쪽 = 1) * (이전오른쪽 = 0) 이면
        가로이동량 -> 1
    (지금위 = 1) * (이전위 = 0) 이면
        회전량 -> 1
    #키 상태를 사용했기 때문에 이전 정보를 새로운 정보로 덮어쓰기
    이전왼쪽 -> 지금왼쪽
    이전오른쪽 -> 지금오른쪽
    이전위 -> 지금위
    #떨어진다
    낙하카운트 -> 낙하카운트 + 1
    낙하카운트 = 10 이면
        세로이동량 -> 1
        낙하카운트 -> 0
    #이동을 반영
    세로 -> 세로 + 세로이동량
    가로 -> 가로 + 가로이동량
    회전 -> 회전 + 회전량
    겹친다?(세로, 가로, 회전, 종류) 이면
        #원래 위치로
        세로 -> 세로 - 세로이동량
        가로 -> 가로 - 가로이동량
```

```
        회전 -> 회전 - 회전량
        (세로이동량 > 0) * (가로이동량 = 0) 이면
            쌓아서열지우기(세로, 가로, 회전, 종류)
            세로 -> 1
            가로 -> 5
            회전 -> 0
            종류 -> 종류 + 1
            종류 > 6 이면
                종류 -> 0
    사각형움직이기(세로, 가로, 회전, 종류)

패턴을미리만들어둔다() 정의
    메모리6개기억시키기(500, 0, -1, 0, 1, 0, 2) #모양0
    메모리6개기억시키기(506, -1, -1, 0, -1, 0, 1) #모양1
    메모리6개기억시키기(512, 0, -1, -1, 0, 0, 1) #모양2
    메모리6개기억시키기(518, 0, -1, -1, 1, 0, 1) #모양3
    메모리6개기억시키기(524, 0, -1, 1, 0, 1, 1) #모양4
    메모리6개기억시키기(530, 0, 1, 1, 0, 1, 1) #모양5
    메모리6개기억시키기(536, 0, 1, 1, -1, 1, 0) #모양6

메모리6개기억시키기(시작번호, 수0, 수1, 수2, 수3, 수4, 수5) 정의
    메모리[시작번호 + 0] -> 수0
    메모리[시작번호 + 1] -> 수1
    메모리[시작번호 + 2] -> 수2
    메모리[시작번호 + 3] -> 수3
    메모리[시작번호 + 4] -> 수4
    메모리[시작번호 + 5] -> 수5

칸위치를계산한다(세로, 가로, 회전, 종류) 정의
    메모리[0] -> 세로
    메모리[1] -> 가로
    시작번호 -> 500 + (종류 * 6)
    횟수 -> 0
    횟수 < 3 동안
        상대세로 -> 메모리[시작번호 + (횟수 * 2) + 0]
        상대가로 -> 메모리[시작번호 + (횟수 * 2) + 1]
        회전횟수 -> 0
        회전횟수 < 회전 동안
            #교체
            교체용 -> 상대세로
            상대세로 -> 상대가로
            상대가로 -> 교체용
            #플러스/마이너스 교체
            상대가로 -> -상대가로
            회전횟수 -> 회전횟수 + 1
        메모리[2 + (횟수 * 2)] -> 세로 + 상대세로
        메모리[3 + (횟수 * 2)] -> 가로 + 상대가로
        횟수 -> 횟수 + 1

겹친다?(세로, 가로, 회전, 종류) 정의
    칸위치를계산한다(세로, 가로, 회전, 종류)
    출력 -> (메모리[100 + (메모리[0] * 12) + 메모리[1]] = 1) +
        (메모리[100 + (메모리[2] * 12) + 메모리[3]] = 1) +
```

```
        (메모리[100 + (메모리[4] * 12) + 메모리[5]] = 1) +
        (메모리[100 + (메모리[6] * 12) + 메모리[7]] = 1)

쌓아서열지우기(세로, 가로, 회전, 종류) 정의
    칸위치계산하기(세로, 가로, 회전, 종류)
    쌓기(메모리[0], 메모리[1])
    쌓기(메모리[2], 메모리[3])
    쌓기(메모리[4], 메모리[5])
    쌓기(메모리[6], 메모리[7])
    지운다()

사각형움직이기(세로, 가로, 회전, 종류) 정의
    칸위치계산하기(세로, 가로, 회전, 종류)
    사각형(메모리[0], 메모리[1], 990000) #빨간색
    사각형(메모리[2], 메모리[3], 990000) #빨간색
    사각형(메모리[4], 메모리[5], 990000) #빨간색
    사각형(메모리[6], 메모리[7], 990000) #빨간색
    메모리[55000] -> 1 #편지
    사각형(메모리[0], 메모리[1], 0) #지우기
    사각형(메모리[2], 메모리[3], 0) #지우기
    사각형(메모리[4], 메모리[5], 0) #지우기
    사각형(메모리[6], 메모리[7], 0) #지우기

쌓기(세로, 가로) 정의
    메모리[100 + (세로 * 12) + 가로] -> 1
    사각형(세로, 가로, 999999)

벽과바닥그리기() 정의
    벽그리기()
    바닥그리기()

벽그리기() 정의
    횟수 -> 0
    횟수 < 20 동안
        쌓기(횟수, 0) #왼쪽
        쌓기(횟수, 11) #오른쪽
        횟수 -> 횟수 + 1

바닥그리기() 정의
    횟수 -> 0
    횟수 < 10 동안
        쌓기(19, 1 + 횟수)
        횟수 -> 횟수 + 1

사각형(세로, 가로, 색) 정의
    세로횟수 -> 0
    세로횟수 < 4 동안
        가로횟수 -> 0
        가로횟수 < 4 동안
            메모리[60000 + (세로 * 500) + (가로 * 5) +
                가로횟수 + (세로횟수 * 100)] -> 색
            가로횟수 -> 가로횟수 + 1
        세로횟수 -> 세로횟수 + 1
```

```
지운다() 정의
    세로 -> 0
    세로 < 19 동안
        지워진다? -> 1
        가로 -> 1
        가로 <= 10 동안
            지워진다? -> 지워진다? * (메모리[100 + (세로 * 12) + 가로] = 1)
            가로 -> 가로 + 1
        지워진다? 이면
            #쌓였다고 기억하고 있는 메모리를 0으로 설정
            가로 -> 1
            가로 <= 10 동안
                메모리[100 + (세로 * 12) + 가로] -> 0 #메모리를 0으로
                사각형(세로, 가로, 0) #검정색으로 칠해서 지우기
                가로 -> 가로 + 1
            옮긴다(세로)
        세로 -> 세로 + 1

옮긴다(지워지는열의세로) 정의
    세로 -> 지워지는열의세로 - 1
    세로 >= 0 동안
        가로 -> 1
        가로 <= 10 동안
            메모리[100 + (세로 * 12) + 가로] = 1 이면
                메모리[100 + (세로 * 12) + 가로] -> 0 #지우기
                사각형(세로, 가로, 0) #검정색으로 칠한다
                쌓기(세로 + 1, 가로)
            가로 -> 가로 + 1
        세로 -> 세로 - 1
```

갑자기 정신이 멍해질 수도 있지만, 어느 정도 양인지 알려주기 위해서 일부러 '게임' 파일에 있는 내용을 전부 실었다. 180행 정도의 양이다. 200자 원고지에는 10행이 있으므로 한 행의 길이를 무시하더라도 20장 정도가 된다. 문자 수로 말하면 3,000자보다 약간 많다. 즉, 180행 3,000자의 '해괴한 문장'을 작성하면 앞서 본 게임을 만들 수 있다. 따라서 여러분은 이 정도 양의 프로그램을 작성하지 않으면 안 된다.

판매용 프로그램의 크기는?

그러면 판매용 프로그램은 어느 정도의 규모인지 보도록 하자. 내가 알고 있는 어떤 스포츠 게임은 2,000개의 파일과 200만 행 정도의 프로그램으로 작성돼 있다. 다른 스포츠 게임은 조금 작은데, 500개의 파일과 50만 행 정도로 구성되어 있기도 하다.

전자는 20명 정도의 프로그래머가 수년에 걸쳐 만든 것이고, 후자는 최대 7명의 프로그래머가 약 1년 반 정도에 걸쳐 만든 것이다. 세상에는 수백억 원을 들여서 만드는 대규모 게임도 있으며,

그 정도 규모이면 30명 또는 50명 정도의 프로그래머가 참여하게 된다. 게다가 윈도우 자체도 프로그램인데, 수천만 행에 달하는 코드로 작성돼 있다고 한다. 이것은 전문가가 다루는 규모다. 여러 사람이 작업한다고 해도 한 사람당 수만 행에 달하는 코드를 작성해야 한다.[11]

180행밖에 되지 않는 예제 게임은 전문 프로그래머라면 빠르면 2시간 만에 만들 수 있다. 나는 게임 전문가가 아니라서 반나절은 걸렸다. 2일 정도 걸린다면 전문가로서는 수준이 낮다고 할 수 있다. 즉, 이 책을 통해서는 전문가가 수 시간에 걸쳐 할 수 있는 정도밖에 배울 수 없다. 하지만 반대로 말하면, 수만 명이나 되는 프로그래머들이 만든 것의 수백 분의 일 정도의 양만 있으면 이 책에서 다루는 게임 정도는 쉽게 만들 수 있다는 것을 의미한다.

자, 어느 정도 규모인지 감을 잡았을 것이다. 이제 겁을 먹을지 용기를 낼지는 여러분에게 달렸다.

1.2 무엇부터 시작할 것인가?

여러분이 달성해야 할 목표가 어느 정도인지 알았을 것이다. 그러면 다시 이전 얘기로 돌아가보자.

지금까지의 설명으로 'Sunaba라는 프로그래밍 언어의 규칙을 따라서 만든 게임이 프로그램이다'라는 것을 알았을 것이다. 프로그램을 작성한다는 것은 Sunaba 규칙을 따라 문장을 만드는 것이다. 여기서 Sunaba 규칙이 쉽게 익힐 수 있는 것이라면 빨리 배우면 된다. 반대로, 그다지 쉽지 않은 규칙이라면 Sunaba 규칙 중 필요한 부분만 선별해서 배우면 된다.

1.2.1 프로그래밍 언어 학습법

결론부터 말하자면, Sunaba의 규칙을 하나씩 차례대로 배우게 된다. 그렇다고 배울 수 없을 정도로 많은 규칙이 존재하는 것도 아니다. Sunaba의 주요 규칙은 종이에 적으면 2페이지 정도밖에 되지 않는다.[12] 하지만 규칙을 모두 배웠다고 해도 아무런 도움이 되지 않는다.

프로그램은 해괴하지만 일반적인 문장과 모습이 비슷하다. 그래서 게임을 문장으로 모두 바꿔서 생각해보도록 하자.

11 우리의 '게임' 파일은 정말 최소한의 코드로 한정한 것이다. 색상도 없고 시작 화면이나 점수 표시 기능도 없다. 이 부분만 추가해도 수천 행의 코드가 만들어진다.

12 부록에 있는 'Sunaba 추가 사양'은 실제 2페이지밖에 되지 않는다.

'문장 규칙만 배우면 문장을 만들 수 있다'라고 할 수 있을까? 그렇지 않다. 우리가 사용하는 언어의 문법이나 단어 사용법 등을 배워야 하지만, 그것만으로는 문장을 만들 수 없다. 마찬가지로, Sunaba의 문법을 배워도 그것으로 문장을 만들 수는 없다. **프로그래밍 언어를 배우는 것과 프로그램 만드는 법을 배우는 것은 전혀 다른 문제다.**

이후로 여러분이 학습하게 되는 순서는 영어처럼 일반적인 언어를 배우는 순서와 거의 같다. 일단, 몇 가지 단어와 최소한의 문법을 배우고 그것을 이용해서 반복 표현한다. 이런 경험을 통해 언어에 익숙해지는 것이다. 'I have a pen.' 정도의 문장부터 시작하게 된다.

그리고 이것이 익숙해져서 좀 더 정확히 표현하고 싶은 문장이 있다면, 필요한 단어나 문법을 조금씩 늘려나간다. 이것을 다시 반복함으로써 해당 표현에 익숙해지는 것이다. 결국은 반복이다. 단어를 잘 모르는 상태에서는 알고 있는 단어만 사용해서 표현해야 하므로 힘이 들 수 있다. 하지만 이런 힘든 과정이 반드시 있어야 한다.

1.2.2 어떤 것부터 만들어야 할까?

'게임을 만들기 위해서 어떻게 Sunaba의 규칙을 배우면 될까?'가 문제가 된다. 하나씩 그 규칙을 배워간다고 앞서 언급했다. 그러면 다음에 생각해야 할 것은 '어떤 것부터 시작할까?'이다.

하지만 아직 어떤 규칙이 있는지조차 모른다. 규칙을 정리한 목록이 있다고 해도 어떤 규칙부터 배워야 할지도 모른다. '명사'나 '형용사'처럼 다양한 문법 요소가 있지만, 어느 것을 사용하는지는 무엇을 할지가 정해지지 않으면 알 수 없다.

따라서 '게임의 어느 부분부터 만들기 시작해야 하는가?'에 대해 생각해야 한다. 그것이 정해지면 '해당 부분을 만드는 데 필요한 Sunaba 규칙'을 선별할 수 있다. 제일 먼저 '양파를 다진다'라고 정했다면, 당연히 '다지기 방법'을 배워야 한다.

1.2.3 만들기 쉬운 요소를 선별할 수는 없는가?

잠시 예제 게임을 가지고 놀면서 생각해보자.

'이 게임을 만드는 데 필요한 것 중 쉽게 손을 댈 수 있는 것'이 무엇인지 생각해서 그것만 별도로 빼낼 수 없을까? 예를 들어, 키보드의 좌우 키를 누르면 빨간색 사각형이 좌우로 움직인다. 좌우가 귀찮으면 오른쪽만 생각해도 좋다. 또는 움직이지 않는 그림이라고 생각하면 어떨까? 이 게임 화면이 나오기 위해서는 화면에 그림을 그려야 한다. '특정 순간의 게임 화면을 만든다'라는

부분만 별도로 생각하면 어떨까? 화면 전체가 귀찮다면 빨간색 사각형만 생각해도 된다. 또한, 빨간색 사각형 4개가 한 세트로 떨어지지만, 4개 세트로 생각하지 않고 한 개의 작은 사각형만 생각할 수도 있다.

무엇을 선택하든 괜찮지만, 별도로 추출할 요소는 만들기 쉬운 것이 좋다. 가능한 적은 학습량으로 달성할 수 있는 요소라면 금상첨화다. 지금은 시작하는 단계다. 갑자기 어려운 것을 만들려고 해도 될 리가 없다.

그렇게 생각한다면 '사각형을 1개 그린다'가 나쁘지 않은 것 같다. '오른쪽 화살표 키를 누르면 빨간색 사각형이 오른쪽으로 이동한다'는 것을 만들기 위해서는 사각형을 먼저 그려야 한다. 즉, 화면에 그림을 그리는 것이 제일 먼저다.

'처음은 사각형을 1개 그리면 어떨까?'가 첫 번째 작업이 된다. 왜 사각형을 그리는 것부터 시작하는지, 그리고 왜 그렇게 정했는지는 어느 정도 이해했으리라 생각한다.

1.3 이번 장에서 알려주고 싶었던 것

이번 장에서 말하고 싶었던 것을 정리하면 다음과 같다.

- Sunaba 프로그램을 실행하는 방법
- 프로그램이라는 것은 특정 규칙에 의해 작성된 '해괴한 문장'이다
- 목표 달성을 위해 먼저 무엇부터 손을 댈지 생각해보자

제일 중요한 것은 마지막 항목이다. 아무리 큰 목표라도 반드시 첫발을 내딛는 과정이 있다. 무언가를 만들 수 있는 상태가 되지 않으면 아무것도 시작할 수 없다. 그리고 그 첫발을 내딛기 위해 무엇을 하면 좋을지는 스스로 생각해야 한다.

다음 장부터는 구체적으로 프로그램을 만들어볼 것이다. 드디어 본격적인 프로그램 세계로 들어가는 것이다. 그러면 지금부터 사각형을 그리는 방법을 알아보도록 하자.