

Final Project: Yacht Dice

과목명: [CSE4116] 임베디드시스템소프트웨어
담당교수: 서강대학교 컴퓨터공학과 박성용
학번 및 이름: 20161566, 권형준
개발 기간: 2021.06.01~2021.06.21

목차

1. 프로젝트 설명

- 1) Yacht Dice
- 2) 실행 화면

2. 프로젝트 흐름도

3. Java Application

- 1) Functions
- 2) Layouts
- 3) JNI
- 4) Resources

4. Device Driver

- 1) FND
- 2) Dot
- 3) Push_Switch

5. System Call

6. 정리

1. 프로젝트 설명

1) Yacht Dice

저는 프로젝트의 주제를 Yacht Dice(야추 게임)로 설정하였습니다. 프로젝트에 대한 본격적인 설명에 앞서, 우선 Yacht Dice가 어떤 게임인지 간단한 규칙과 플레이 방법에 대해 설명하겠습니다.

Yacht Dice는 총 5개의 정육면체 주사위를 가지고 하는 게임입니다. 주사위를 굴려서 나오는 숫자의 조합을 <그림 1>의 족보와 비교하여 점수를 얻는 방식으로 진행이 됩니다. 1:1로 진행이 되며, 각자 돌아가면서 총 12번의 기회를 얻게 됩니다. 한번의 기회당 최대 3번 다시 주사위를 던질 수 있으며, 5개 중 부분 적으로(2개만, 3개만) 던질 수 있습니다. 총 12번의 기회가 끝났을 때 더 점수가 높은 사람이 이기게 됩니다.

Yacht Dice을 하기 위해 꼭 알아야되는 족보에 대해서 간단히 설명 드리겠습니다.

- ① <그림 1>의 "Aces~Sixes"는 5개의 주사위 중 (현재 항목) X (현재 항목의 주사위 수)로 점수가 구성이 됩니다. 예를 들어 5개 주사위가 각각 [5, 4, 5, 2, 1]이 나왔다고 한다면, Aces의 점수는 $1 \times 1 = 1$, Duces: $2 \times 1 = 2$, ..., Fives: $5 \times 2 = 10$ 이 됩니다. Aces~Sixes까지의 점수 합이 63을 넘게 되면 최종 점수에 35점의 보너스 점수를 받게 됩니다.
 - ② "Choice"항목은 현재 5개 주사위의 합을 의미합니다.
 - ③ "4 of a Kind"는 현재 5개의 주사위 중 4개의 주사위가 같은 숫자를 보이고 있는 경우에, 전체 5개 주사위의 합이 점수가 됩니다. 예를 들어서 (4,4,4,4,5)가 나왔다면 21이 점수가 됩니다.
 - ④ "Full House"는 5개의 주사위(n_1, n_2, n_3, n_4, n_5)중 ($n_1 == n_2, n_3 == n_4 == n_5$)를 만족하는 경우에 5개 주사위의 합이 점수가 됩니다.
 - ⑤ "S.Straight"은 Small Straight의 약자로 4개의 연속된 숫자가 나온다면 15점을 얻습니다.
 - ⑥ "L.Straight"은 Large Straight의 약자로 5개의 연속된 숫자가 나온다면 30점을 얻습니다.
 - ⑦ 마지막 "Yacht"는 5개의 같은 숫자가 나오면 50점을 얻습니다.
- 매 차례 마다 12개의 항목 중 하나만 채울 수 있으며, 이미 채운 항목은 이후에 중복으로 채울 수 없습니다. 따라서 총 12번의 기회가 지나가면 모든 항목이 꽉 차게 됩니다.

Turn 1/12		
Categories		
Aces	■	
Deuces	■	
Threes	■	
Fours	■	
Fives	■	
Sixes	■	
Subtotal	0/63	0/63
+35 Bonus		
Bonus if ■ - ■ are over 63 points		
Choice	■	
4 of a Kind	■	
Full House		
S. Straight	■	
L. Straight	■	
Yacht		
Total	0	0

<그림 1>

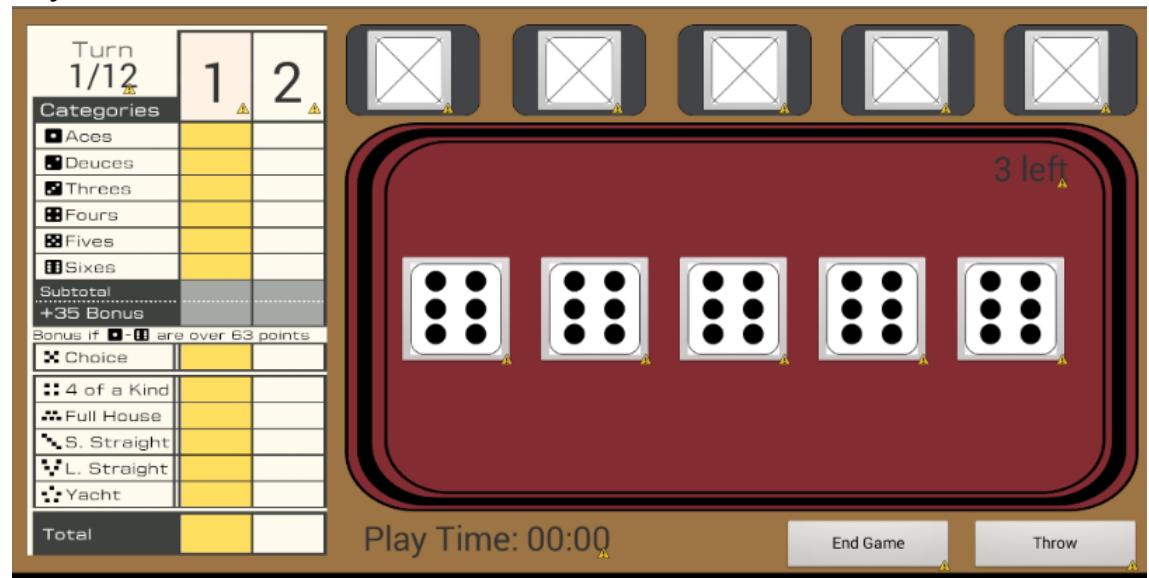
2) 실행 화면

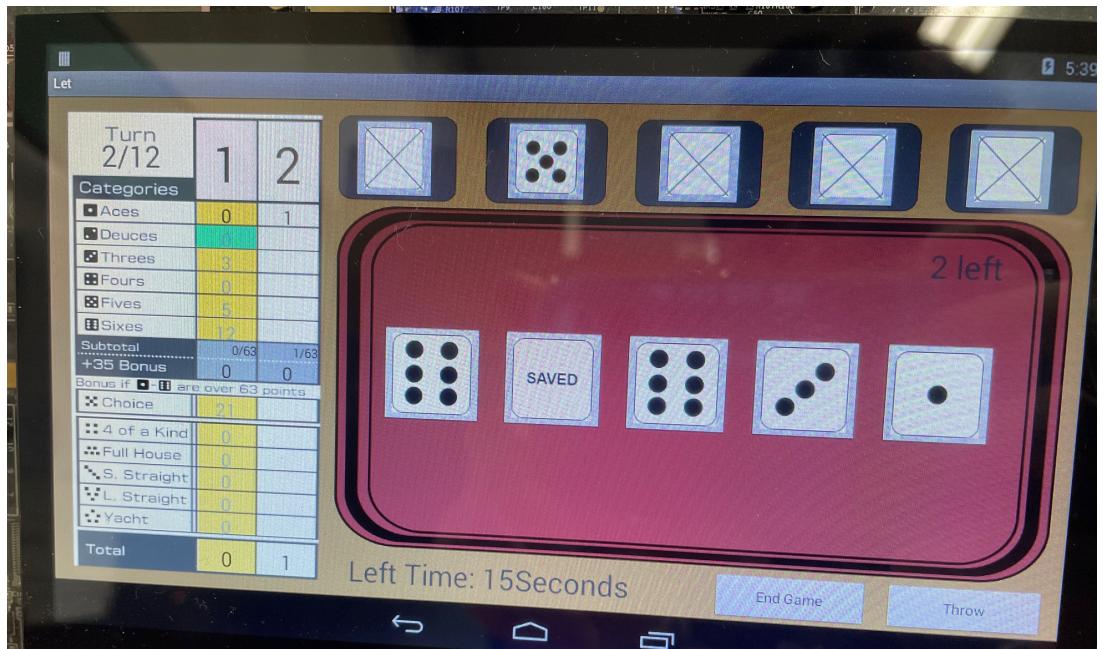
- 시작 화면



Play!	Play 화면으로 넘어갑니다.
Music On/Off	시작과 동시에 Theme Song이 흘러나온다. 조용한 환경에서 게임을 하고 싶은 경우 이를 눌러 음악을 끄고 켤 수 있습니다.
End Game	Game을 종료하고 바탕화면으로 나갑니다.

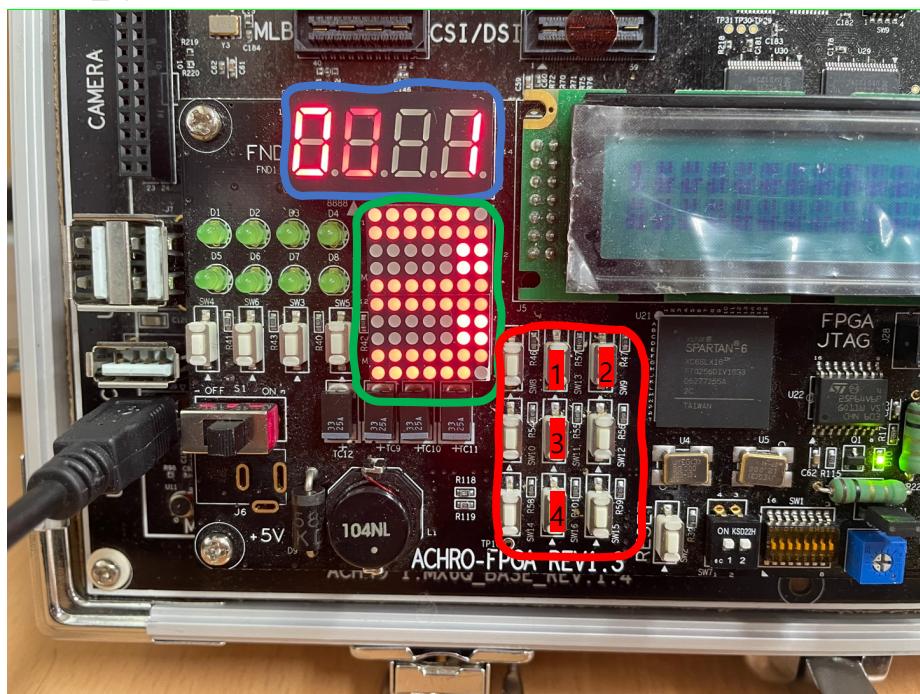
- Play 화면





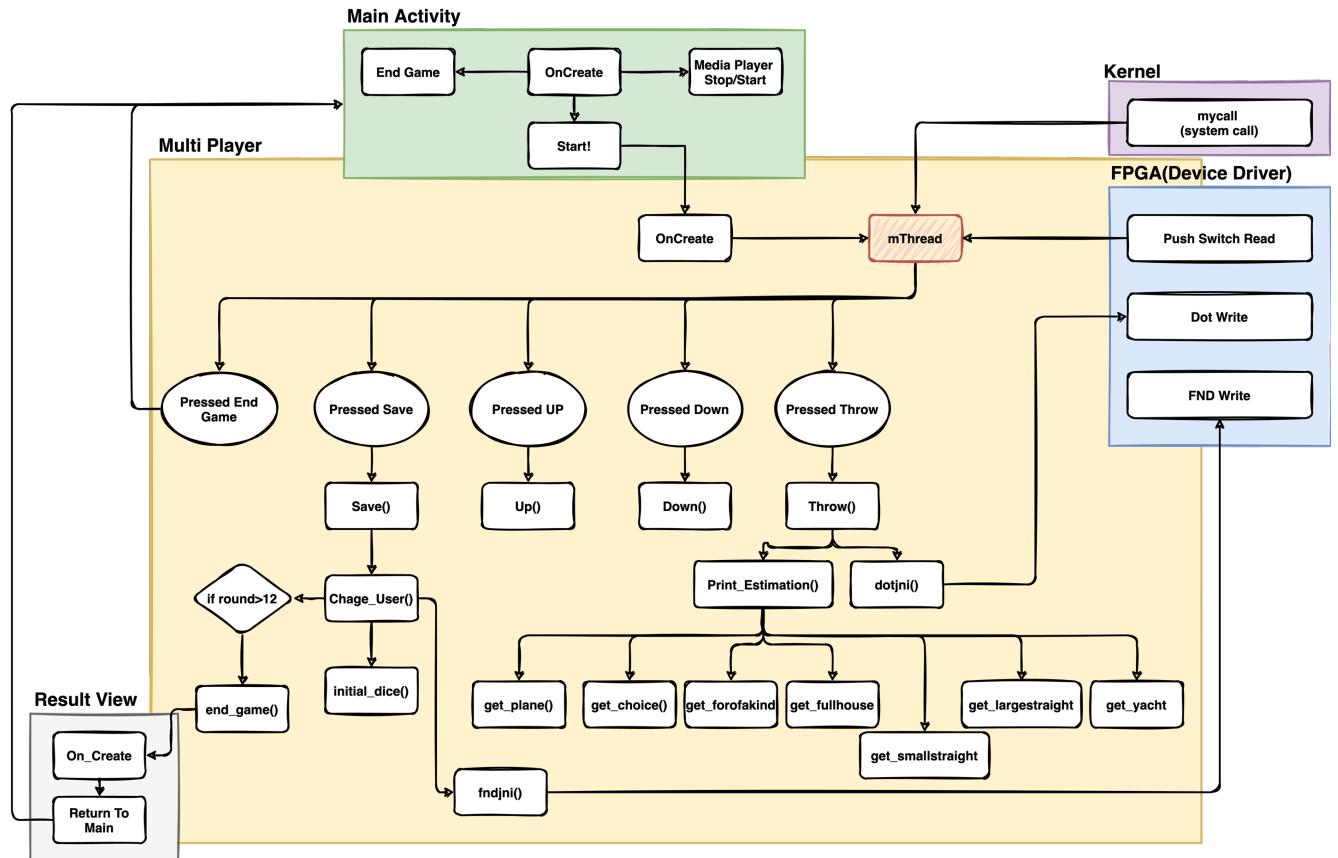
	한번의 차례에서 총 3번 주사위를 돌릴 수 있으며, 새로 던지고 싶지 않은 주사위는 눌러서 위에 보관함에 넣을 수 있습니다. 이 기호는 보관함을 뜻합니다.
	육면체 주사위를 뜻하며, 보관하고 싶으면 누를 수 있습니다. <Image Button>으로 구현하였습니다.
	아직 던지지 않은 주사위는 '?'로 표시가 됩니다. 새로운 게임이 시작된 경우, 다음 Player에게 차례가 넘어간 경우에 이런 주사위 표시가 나타납니다.
	위에서 육면체 주사위를 눌러서 저장을 한 경우, 기존의 주사위가 있던 자리에는 "SAVED" 표시가 됩니다.
	현재 Player가 이번 차례에 던질 수 있는 남은 횟수입니다.
	주사위를 던집니다. SAVED 표시가 된 주사위는 던져지지 않습니다.
	게임을 종료하고 Main 화면으로 돌아갑니다.
	현재 Player에게 남은 시간을 알려줍니다. 한번 주사위를 던진 이후 30초 이내로 결정을 해야 됩니다.
	현재 Target을 표시합니다. Target이란 족보에 넣을 수 있는 자리 (최대 12개)에서 이번 차례에 저장할 곳을 뜻합니다. 이는 FPGA의 Push Switch를 통하여 위아래로 조절할 수 있습니다.
진한 글씨, 연한 글씨	진한 글씨는 현재 저장한 항목을 뜻하며, 연한 글씨는 현재 주사위 상태를 바탕으로 해당 항목에서 얻을 수 있는 점수를 한눈에 보기 편하게 나타낸 것입니다.

- FPGA 설명



FND	현재 Player의 Total Score를 표시해줍니다.
DOT	현재 던진 주사위 중 가장 많이 나온 숫자를 표시해줍니다. 만약 Tie가 있다면 '?'를 출력합니다.
Push_Switch	<p>스위치를 통하여 앞에서 설명한 Target도 조정할 수 있으며 Target에 save도 할 수 있습니다. 또한 Throw또한 Switch를 통하여 할 수 있습니다.</p> <p>1: 위 2: 주사위 던지기 3: 저장 4: 아래로</p>

2. 프로젝트 흐름도



위는 전체적인 프로젝트의 흐름도 입니다. 색으로 표시한 영역은 Activity이거나 Kernel영역 또는 Device Driver영역입니다. 주요 기능들은 MultiPlayer Activity에 구성이 되어 있습니다. Player 입력을 Button Onclick() 또는 mThread를 통해서 받고 이에 따른 함수 수행으로 전체적 흐름이 구성됩니다. mThread는 FPGA(Device Driver)에서 값을 읽어오거나, System Call을 통해 값을 읽어온 것을 application level에 전달해주기 위해 background에서 돌고 있습니다. 아래에서 프로젝트의 요구사항대로 크게 3가지로 나누어 설명을 진행하도록 하겠습니다.

3. Java Application

1) Functions

① Handler & BackThread

JNI를 통해서 Device Driver와 통신을 하기 위한 방법 2가지 중 저는 Thread를 따로 만들어서 Polling을 하는 방식으로 구현하였습니다. 실습 코드에서 주어진 것을 바탕으로 Thread를 extend하여 BackThread를 만들고, 해당 Thread에서 View에 접근할 수 있게 하기 위해 Handler를 선언하였습니다. 0.3초 단위로 “readjni(), gettime()”라는 native함수를 호출하게 되는데, 이는 각각 어떤 switch가 눌렸는지, 보드의 현재 시간을 받아오는 기능을 수행합니다. 이에 대한 설명은 뒤에 JNI에서 설명하겠습니다. 버튼은 총 4가지를 입력받을 수 있으며 이에 대한 설명은 위에 FPGA설명에 있습니다. ‘위’를 누르면 UP()함수가, ‘아래’를 누르면 DOWN()함수가, ‘save’를 누르면 SAVE(), Change_User가 호출되며, Throw는 화면 내의 button과 fpga의 switch들 다 구현하였습니다.

② Print_Raw & Print_Estimation & clear

Print와 Clear함수들은 화면의 점수판에 출력을 하기 위한 함수입니다. Text가 들어갈 수 있는 영역을 TextView로 Layout에서 설정하였고, 해당 부분의 ID를 현재 Dice, saved_dice등등의 변수에 저장된 값을 토대로 수정하였습니다. Print_Estimation은 게임을 더 쉽게 하기 위하여 현재 주사위를 토대로 해당 항목에서 얻을 수 있는 점수를 연한 글씨로 표시하였습니다.

③ Save & Change_User

Save, Change_User는 같이 세트인 함수로 Player가 현재 항목에 점수를 쓰려고 할 때 호출이 됩니다. 현재 항목의 점수를 저장하고 점수판 수정, FPGA 수정 등등의 다른 함수들이 호출된 이후 다음 Player에게 기회를 넘기게 됩니다. 30초의 제한시간이 지나가게 되면 자동으로 현재 항목에 점수가 저장이 된 다음 Player에게 기회가 넘어가도록 구현하였습니다. Timer는 Application단에서 구현하지 않고 System call로 현재 kernel의 시간을 polling해 와서 비교하는 방식으로 구현하였습니다. 이에 대한 설명은 밑에 System Call에서 하겠습니다.

④ UP & DOWN

사용자가 점수를 기록하고 싶은 항목으로 이동할 때 호출이 됩니다. Layout상에서 현재 선택한 항목의 TextView의 Background color를 바꿔줌으로써 focusing을 조절하였습니다. 이미 저장된 항목은 뛰어 넘으며 가장 아래에서 아래를 누른 겨우, 가장 위에서 위를 누른 경우에는 한바퀴 돌아서 가장 위로 그리고 가장 아래로 다시 목표가 움직이도록 구현하였습니다.

2) Layouts

① Relative Layout

다양한 위치에 이미지 밑 text를 삽입하기 위하여 relative layout을 사용하였습니다. 하지만 오래된 버전으로 인해 UI설정하는 과정이 불편하여 결국 각 View의 좌표를 수동으로 설정해주었습니다.

② Image Button

주사위를 저장하거나 저장된 주사위를 버리기 위해서 주사위 그림을 Image Button으로 구현하였습니다.

3) JNI

① Readjni

Device Driver를 사용하기 위한 native function입니다. "/dev/fpga_push_switch" Device Driver를 open하여 read를 하게 됩니다. 이 Device Driver는 fpga보드에서 push_switch에 접근하기 위해 실습에서 사용한 device driver를 그대로 사용하였습니다. 이때 읽어야 되는 버튼 총 4가지를 구분하여 이에 해당하는 값을 return해주었습니다.

② Fndjni

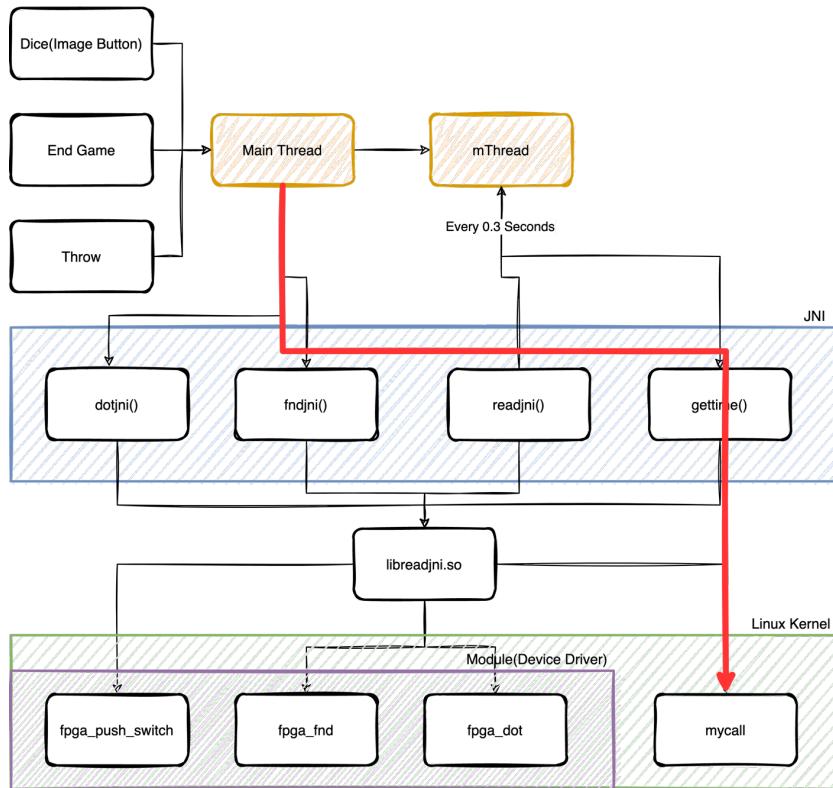
FND device에 출력을 하기 위해 "/dev/fpga_fnd"를 Open하고 write을 하는 native function입니다. Fnd device driver 또한 실습에서 주어진 것을 그대로 사용하였습니다. FND에는 현재 Player의 총 점수를 출력해주도록 사용하였습니다.

③ Dotjni

Dot Device에 출력을 하기 위해 "dev/fpga_dot"를 open하고 write하는 native function입니다. Dot Device Driver의 경우 약간의 수정을 하였습니다. 넘기는 값이 배열이 아닌 숫자로, device driver에서 숫자를 토대로 해당하는 숫자를 출력하게 하였고, 0이 입력되는 경우에는 "?"를 출력하도록 하였습니다. Dot Device는 현재 던진 주사위들 중 가장 많이 나온 숫자를 출력하며, 동률이 있는 경우에는 "?"를 출력하게 했습니다.

④ Gettime

해당 Native function은 System call을 사용하기 위한 함수입니다. 새롭게 추가한 "mycall"이라는 system call의 반환 값을 반환해줍니다. 해당 system call은 현재 Kernel의 시간을 받아오며, 이를 주사위를 던진 시점을 기준으로 30초 시간을 측정하여 30초가 지나면 차례가 넘어가도록 하였습니다. Application Level에서도 구현이 가능하지만, system call을 사용하기 위해 추가하였습니다.



4) Resources

- ① Images
<실행화면 설명 참고>
- ② Audio
닌텐도사의 '51가지 세계 게임'에 나오는 Yacht Dice의 bgm을 그대로 가져왔습니다. 또한 가장 높은 점수를 주는 Yacht가 나온 경우 추가적인 효과음을 넣었습니다. Audio는 처음 시작 화면에서 자동 시작이 되며, 버튼을 통해 재생과 정지를 설정할 수 있습니다.

4. Device Driver

1) FND

① 사용 목적

현재 Player의 점수를 출력하기 위하여 사용하였습니다. 물론 화면에서도 나오지만 게임의 결과에 직접적으로 영향을 끼치는 점수를 한번 더 강조하기 위해 추가하였습니다.

② 기능

Device에 Write을 하게 되면 Copy_from_user를 통하여 4개의 value를 읽어온다음 이를 device open시에 mapping된 주소로 정해진 형식에 따라 출력함으로써 FND Device에 원하는 숫자를 출력해줍니다. 이는 실습에서 제공되었던 Device Driver를 그대로 사용하였습니다.

2) Dot

① 사용 목적

현재 나와있는 총 5개 주사위 중 가장 많이 나온 숫자를 출력해줍니다. 만약 동률이 있다면 "?"를 출력합니다. 주사위를 저장하기도 하고 정렬된 상태로 표시해주지 않기 때문에 빠르게 Player에게 가장 많이 나온 주사위를 정리하여 보여줌으로써 플레이에 도움을 주도록 설정하였습니다.

② 기능

Dot Device Driver의 경우 실습에서 주어진 device driver에 약간의 수정을 하였습니다. Write시에 넘기는 인자는 크기가 10인 배열이 아닌 const char 하나만 받도록 수정하였습니다. 앱에서 출력하는 결과는 '1'~'6', '?' 뿐이기 때문에 0을 '?'에 Mapping하였습니다. 또한 미리 출력할 배열들을 device driver에 선언하여 device driver에 write하는 함수를 조금 더 간단하게 만들었습니다.

3) Push_Switch

① 사용 목적

점수판에서 위, 아래로 움직이기 위한 컨트롤러와 점수를 저장하고 주사위를 새로 던지기 위해 push_switch를 사용하였습니다. 0~8까지로 구분 지을 수 있으며, 1은 '위' 2는 '주사위 새로 던지기', 4는 '저장', 7은 '아래'로 mapping하였습니다.

② 기능

Application level에서 이미 주기적으로 polling을 하고 있기 때문에, read가 아무리 non-blocking이더라도 적당히 오래 누른다면 인식이 되도록 하였습니다. 실습에서 주어진 Device Driver를 그대로 사용하였습니다.

5. System Call

① 사용 목적

Yacht Dice는 단순히 운으로 하는 게임 같지만, 은근히 전략과 확률적 요소가 많이 들어갑니다. 따라서 각자의 Player시간을 제한하지 않는다면 게임이 길어질 수 있습니다. 이를 위해 Timer는 필수적인 요소였습니다. System Call에서 현재 시간을 가져와 30초 시간 인터벌을 정하고 계속 System Call을 주기적으로 호출하여 30초가 지났는지 확인하였습니다.

② 기능

수업시간에 배운 <linux/time.h>에 정의되어 있는 timeval구조체를 사용하여, do_gettimeofday를 통해 현재 시간을 초 단위로 받아왔습니다. 그리고 이를 반환하였습니다.

6. 정리

전체 프로젝트에 대한 노력을 100이라고 한다면 Android Application에 80, Device Driver에 10, System Call에 10정도 투자한 것 같습니다. 아무래도 대부분의 기능이 벌써 Application 단에서 처리가 가능하고 UI에 무겁게 의존하는 게임(그렇다고 높은 성능이 필요하지는 않은)의 특성상 low level의 함수들은 필수적인 요소가 아니었던 것 같습니다. 그래도 Android Application에서부터 출발하여 linux kernel의 system call까지 내려가는 경로와 그 사이에 거치는 level들에 대한 복습을 할 수 있어서 의미 있었던 것 같습니다.