|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **교과목 명** | | 전자HW설계 | | | | | | | |
| **설계 제목** | | Audio 제어를 통한 장애물 피하기 | | | | | | | |
| **설계 기간** | | 2018년도 2학기 | | | | | | | |
| **지도교수** | | 이재환 교수님 | | | | | | | |
| **팀원** | | **이름** | 오승현 | **학번** | 2014122165 | **☎** | 010-7247-0540 | **E-mail** | william20000@naver.com |
| **이름** | 이지민 | **학번** | 2015124172 | **☎** | 010-6513-3223 | **E-mail** | ljm00111@gmail.com |
| **이름** |  | **학번** |  | **☎** |  | **E-mail** |  |
| **목표설정** | **설계 목표** | 설계의 최종 목표 :  1. HW 설계 과목에서 지금까지 배웠던 interrupt handling, timer/counter 사용, pushbutton, switch 등의 i/o 사용법을 응용하고 추가적으로 필요한 peripheral을 사용하여 C language로 프로그램을 구현하도록 함.  2. 우리 조가 만들 프로그램은 기본적으로 VGA에 표시된 장애물을 피하는 video game이다. 그러나 key를 이용하여 개체의 움직임을 제어하는 일반적인 video game과는 달리, 우리 조는 mic in port와 audio-in/out port를 사용하기 위하여 소리를 input으로 받는 game을 만들 것이다. mic를 통해 입력된 소리의 음량, 즉 소리의 진폭의 크기에 따라 개체가 세로 방향으로 얼마나 움직일지를 판단하는 알고리즘을 작성하고, 개체의 움직임과 장애물을 VGA에 display한다. 장애물을 피하기 위해 user는 음량을 조절하며 mic에 목소리를 입력하여 플레이한다.  3. 현재 플레이 중인 round의 점수, 현재까지 플레이 한 round 중의 최고 점수, 현재 플레이 하는 round의 난이도를 HEX5-0까지 display 할 것이다. 난이도는 switch1-0을 사용하여 3개의 레벨을 user가 지정할 수 있도록 한다. Switch가 00이면 EASY MODE, 01이면 NORMAL MODE, 10이면 HARD MODE, 11이면 HARD MODE로 난이도의 기준은 장애물의 접근 속도를 기반으로 한다. 현재까지 플레이 한 round 중의 최고 점수는 HEX5-4에 표시하고, 현재 플레이 중인 round의 점수는 실시간으로 HEX3-2에, 현재 플레이 하는 round의 난이도는 HEX0에 1-4의 수로 표현한다.  4. 현재 플레이 중인 round의 점수는 실시간으로 계산하여 HEX3-2에 디스플레이 할 것이다. 장애물이 VGA의 왼쪽 벽에 닿았다는 것은, user가 소리를 통해 제어하는 개체가 장애물을 피했음을 의미한다. 그러므로 왼쪽 벽면에 닿는 장애물의 개수를 counter를 이용해 실시간으로 세어 점수를 1점씩 증가 시키도록 한다.  5. push button은 KEY0, KEY2, KEY3를 사용할 것이다. KEY0는 일정 시간동안 fever mode를 가동하는 버튼이다. 플레이 중 특정 모양의 아이템이 나타나는데, 이는 미리 설정한 유효 시간 동안은 장애물을 피하지 않아도 game이 stop 되지 않고 점수가 올라가는 fever mode를 가동하는 아이템이다. 게임을 진행하는 동안 개체 쪽으로 접근해 오는 아이템을 획득하게 되면, 이는 KEY0가 눌릴 때까지 사용을 보류할 수 있다. KEY0를 누르면 아이템을 사용한 것이므로 즉시 fever mode가 동작한다. 이 때 fever mode가 유효한 시간을 일정하게 제한하기 위해서 timer가 필요하다. key가 눌린 이후에 timer를 start하여 5초를 세고, timer interrupt가 들어오면 fever mode를 종료하도록 한다.  6. KEY2는 지금까지의 최고 점수가 기록된 HEX를 0으로 클리어하는 버튼이다. 이 버튼을 누르면 지금까지 기록된 최고 점수는 0으로 리셋하고 다시 플레이하여 새로운 점수로 최고 점수를 기록할 수 있다.  KEY3는 게임을 재시작 하는 버튼인데, 게임을 플레이하다 장애물에 충돌하여 game over가 되었을 경우 화면에 user 개체만 display 되어 있을 것이다. 이 때 KEY3를 누르는 즉시 게임을 처음부터 재시작 한다. | | | | | | | |
| **설계 규격** | 본 프로젝트는 DE1\_SoC 보드를 사용하여 설계를 진행한다. Altera의 DE1\_SoC FPGA를 사용하기 위해 보드의 processor는 nios2 Architecture를 사용한다. 간단히 DE1-SoC 보드를 살펴보면 크게 Cyclone V HPS부분과 Cyclone V FPGA부분으로 나뉘는데, 이번학기에서 주로 다룬 내용은 FPGA부분이므로, HPS 부분은 설명하지 않겠다. FPGA부분을 살펴보면 LEDR, HEX, SW, KEY, SDRAM, Audio In/Out 등등이 있다.  여러가지 ports중 이번에 사용할 port는 Audio In/Out과, LEDR, HEX, SW, KEY, Clock generator, VGA이다. 각각의 LEDR과 HEX와 Switch는 memory assigned mapped된 FPGA port들의 Base address에서 값을 읽고 쓰는 방식으로 Input과 Output 해주도록 한다. LEDR로는 fever모드의 남은 시간을 display해주고, HEX는 최고점수, 현재점수, 난이도 level을 display하도록 해준다. Switch로는 값을 읽어와서 난이도를 조절해주도록 한다.  KEY와 Clock generator는 controller register를 통해, interrupt를 control 해주도록 한다. 동작하는 중에 interrupt가 발생하면 CPU가 exception handler를 통해 해당 ISR로 가고, interrupt가 KEY[0]이면, fever mode를 시작하고, 시간을 세기 시작하면서 설정해 준 시간을 다 세면 TIMER interrupt가 발생하면서 TIMER를 꺼주고 fever mode를 종료한다. KEY[2] interrupt가 들어오면 memory를 clear해줘서 현재까지의 기록을 초기화해주도록 한다. 게임이 종료된 상태일 때, KEY[3] interrupt가 들어오면 게임을 다시 재시작 할 수 있도록 한다.  Audio In/Out은 소리의 analog값을 Polled I/O 방식으로 가져와서 Audio Base address에서 control해주고, FIFO data들을 right channel(혹은 left channel, 둘이 들어오는 신호는 같을 것이므로)에서 읽어와서 일정 주기 간격으로 sampling된 data들의 평균값을 가져온다. 그 평균값에 따라서 VGA를 통해 display되는 User Character를 제어할 수 있도록 한다.  HEX3\_HEX0의 BASE address는 0xFF200020 HEX5\_HEX4의 BASE address는 0xFF200030이다. 이 Base address에 해당하는 메모리에 값을 넣으면 각각 1비트씩 HEX에 display할 수 있다.  Switch Base address는 0xFF200040이다. 해당 address의 memory값을 읽으면 SW의 값을 읽어올 수 있다.  LEDR의 base address는 0xFF200000이고, 뒤에서부터 1비트씩 write해주면 LEDR가 켜지고 꺼지는 것을 control할 수 있다.  KEY의 Base address는 0xFF200050이고, Base address는 data의 현재 상태를 보여준다. Base address+2는 interrupt mask bit로, KEY의 몇 번째를 interrupt로 받을 것인지 해당하는 비트에 1을 넣어 줌으로써 설정해줄 수 있다. Base address+3은 KEY interrupt가 들어왔을 때, 몇 번째에서 들어왔는지 확인할 수 있고, 다시 임의의 값을 write해주면 Base address는 초기화된다.  TIMER의 Base address는 0xFF202000이고, Base address+1에 bit0에 1을 넣어주면 interrupt가 가능하게 해주고, bit1에 1을 넣어주면 타이머가 계속 돌게 해주고, bit2에 1을 넣으면 start, bit3에 1을 넣으면 stop이 된다. interrupt는 TIMER가 숫자를 다 세면 Base address의 bit0가 1이 되고 interrupt가 발생한다. bit0는 초기화를 해주지 않으면 그대로 1이 유지된다.    Audio\_I/O의 Base address는 0xFF203040이고, WE, RE비트는 1을 넣으면 write와 read interrupt가 enable하게 하고, 각각 interrupt가 발생하면 WI, RI에 1이 입력된다. Base address+2와 Base address+3은 read, write 둘다 가능하고, 각각의 값들은 sampling한 값의 진폭의 data이다.    PIXEL Buffer Controller의 address는 0xFF203020이고, Status register에 있는 bit0을 읽었는데 0이 써져 있으면 Swap을 완료했다는 의미이다. bit1에 1 혹은 0을 넣음으로써 Addressing mode를 선택할 수 있다. S비트는 V-Sync를 맞출 때 이 비트를 확인하고, 0이 들어올 때까지 계속 기다리는 방법으로 Sync를 맞출 수 있다. | | | | | | | |
| **합성/분석** | **관련 기술** | (1) Audio part  DE1-SoC 보드에는 Wolfson WM8731 오디오 코덱 chip이 내장되어 있다. 이는 8kHz부터 96kHz 사이의 주파수 대역과 호환되는 microphone-in, line-in, line-out port를 가지고 있다. digital Audio input는 16~32bit의 길이를 갖고 있고, sampling rate는 8kHz부터 96kHz 사이의 값으로 sampling하게 된다. 각 input의 data는 2's compliment 형태로 소리의 진폭의 크기와 부호를 나타내고, 음수일 때에는 MSB가 1, 양수일 때에는 MSB가 0이 된다. 주파수를 측정하는 방법은 96kHz주기로 sampling된다고 할 때, 96k만큼의 data를 읽어와서, data들이 양수이다가 음수일 때, 그리고 음수이다가 양수일 때 counter의 값을 1씩 증가시키면 주파수를 측정할 수 있다. 이때 주파수 f = (counter의 값 / 2)이다. recording technology는 memory를 적당한 크기만큼 malloc을 통해 동적 할당 해준 후, input channel fifo data를 코덱의 clock에 synchronized하게 메모리 버퍼에 저장하면서 저장하는 memory address를 1씩 증가시킨다. 반대로 play back technology는 sampling rate에 synchronized하게 memory의 base address값부터 읽어와서 Audio fifo data에 넣어주면서 읽어오는 memory address를 1씩 증가시킨다. 이 때, fifo data는 left와 right 두개의 channel이 있어서 두 channel에 모두 넣어준다.  (2) VGA part  기본적인 DE1-SoC의 VGA controller는 640\*480의 화면을 control한다. 화면은 pixel buffer와 character buffer에서 파생된 VGA controller에 의해 display된다. pixel buffer는 RGB 각각의 intensity를 결정한다. 16을 3으로 나눌 순 없어서, R과 B는 5비트, G는 6비트를 사용한다.  V-Sync  controller를 통해 화면에 1/60초마다 주사를 하고 있는데, 이 때 data의 값들이 한 frame을 다 그리기 전인 1/60초 전에 바뀌게 된다면 display되는 화면은 그 다음에 display할 화면을 같이 display하게 될 것이다. 그렇기 때문에 V-Sync technology를 통해 다음 화면을 display하지 못하도록 synchronized하게 display해야 한다. 이 tech를 사용하게 되면 화면의 프레임률은 낮아지지만, 화면이 찢어지는 현상(Tearing)은 없앨 수 있다.    < 사진 인용 : KAU\_전자HW설계\_Lab8 Lecture document(이재환) >  Double buffering  V-Sync를 이용하여 한 프레임을 Synch하게 보내면 화면의 찢어짐 현상은 막을 수 있지만, 화면의 깜빡임(Flickering)을 없앨 수는 없다. Flickering 현상을 없애기 위해서 double buffering 기법을 사용하는데, 원리는 처음에 original buffer에 있는 값을 back buffer에 넣어 놓는다. 다음 frame에서 data가 original buffer에 다 채워지면 original buffer와 back buffer의 값을 swap한다. 그리고 original buffer의 값을 draw하게 되면 결과적으로 이전 frame의 data를 draw하게 된다. 실생활에서도 display방식은 Double buffering 방식을 제일 많이 쓴다.    < 사진 인용 : KAU\_전자HW설계\_Lab8 Lecture document(이재환) > | | | | | | | |
| 제작 | 설계 계획 | 위의 사진은 우리 조가 설계한 프로그램의 각 i/o의 interrupt에 대한 동작을 알고리즘으로 만든 것이다. 프로그램은 코드가 로드 된 이후로 계속해서 timer 혹은 pushbutton의 interrupt를 기다린다.  Interrupt가 들어오면 exception handler를 통해 ipending을 읽어 오는데, 들어온 interrupt가 pushbutton이면 ipending 값이 2가 읽힐 것이고, 어느 key가 눌린 것인지 KEY\_ISR를 통해 edgecapture bit를 확인하는 과정을 거친다.  KEY3가 1이면 game이 시작되고, 이 때 Switch\_BASE 값을 읽어와 switch[1:0] 값에 따라 game의 난이도가 결정된다. Swtich 값이 00이면 EASY MODE, 01이면 NORMAL MODE, 10이면 HARD MODE, 11이면 VERY HARD MODE로 진행한다. 난이도는 장애물이 user 개체로 접근하는 속도를 결정하는데, 이는 switch 신호의 값을 곱하는 등의 연산으로 속도를 결정할 것이다. 이에 대한 설명은 다음 알고리즘에 기술되어 있다. KEY2가 1이면 메모리의 값과 최고 점수를 0으로 리셋한다. 최고 점수가 저장된 변수의 7 segment bit pattern을 결정하고 이를 HEX5-4의 포인터에 저장하여 이를 display함에 따라 최고 점수가 0으로 리셋 되면 자동으로 HEX5-4도 00을 display 할 것이다. KEY0 interrupt가 들어오면 fever mode가 시작된다. fever모드는 아이템을 먹어서 한 개 이상 보유하고 있으면 동작하도록 설계한다. 아이템을 먹게 되면 Counter가 지금까지 먹은 아이템 개수에 1을 더하도록 한다. 5초동안 fever mode를 유지할 것이므로 TIMER의 START bit를 1로 만들어 준다. 단, CONT bit는 0으로 set 해 주어 KEY0가 press 된 순간부터 5초를 세고 timer는 stop 될 수 있도록 한다. Timer는 fever mode에만 쓰일 것이므로 5초를 세도록 timer value를 5s∙100MHz = 500MHz 로 한다. Game을 play 하는 동안에도 fever mode를 가동하는 key에서 interrupt 신호나 Clear해주는 KEY에서 interrupt가 들어오면 해당 기능을 수행할 수 있도록 설계한다.    • 게임의 Flow chart  main에서 계속 반복문을 실행하다가 KEY3 interrupt가 발생하면 handler를 통해 KEY ISR을 거쳐서 game을 start하도록 한다. Switch Base address를 통해 SW의 값을 읽어오고, 그것을 통해 game의 난이도를 조절한다. 난이도에 따라 장애물이 접근하는 속도가 다르도록 설계하고, 장애물은 맨 오른쪽부터 생성되어 왼쪽으로 움직인다. 장애물은 매 1/60초마다 speed 크기의 PIXEL만큼 움직이고, User는 소리의 크기에 따라 위로, 혹은 아래로 움직인다. 이후 V-Sync를 기다려준 후, back buffer를 사용하여 swap해준 후, VGA를 통해 display한다. 그리고 User가 장애물에 닿았는지 판단하고, 닿았으면 게임을 끝내는데, 끝내기 전 현재 점수가 최고 점수보다 높은지 판단하여 최고점수를 리뉴얼 할 지, 그대로 유지할지 판단한다. 만약 장애물에 닿지 않았을 경우에는 장애물이 왼쪽 벽에 닿았는지 판단하고 (닿았으면 User가 장애물을 피한 것임) 닿았으면 현재 점수에 1점을 더해서 Count해준다. 그 값은 HEX에 display해준다. 만약 장애물이 왼쪽 벽에 닿았으면 화면에서 그 장애물은 지워준다. 그리고 새로운 장애물을 다시 생성한다.  • Audio의 input을 읽어오는 부분  Line In에 MIC를 꽂은 후 값들을 Polled I/O방식으로 읽을 것이다. 많지 않은 일정 개수의 sample들(delta x)을 일정 주기간격으로 sampling하여 소리가 가장 큰 sample의 값에 따라 User가 위로 빠르게 혹은 천천히 움직이거나 아래로 떨어지도록 설계한다. VGA를 통해 display할 때, 다음 frame에서의 User의 위치가 위에 있을지, 아래에 있을지 정하는 코드를 C로 작성한다. Tearing과 Flickering 현상을 제거해준 후 display하도록 한다.  Audio의 input이 적절하게 들어오고 있는지를 확인하기 위해 monitor program에 printf()를 통해 주기적으로 Audio의 input값의 최대값을 출력한다. TIMER interrupt를 통해서, 혹은 polled I/O방식을 통해서 값을 출력할 수 있도록 한다.  • User가 장애물에 닿았는지 판단하는 부분  User의 character shape와 장애물 모양이 사각형(x1, y1, x2, y2) 이라면(이 때, x2>x1, y2> y1), (장애물의 크기가 User보다 크다는 조건) , User의 x축이 장애물과 닿는 경우가 있고, y축과 장애물과 닿는 경우가 있다. x축과 닿았을 때에는 User의 y2가 장애물의 y2와 y1사이에 있거나, User의 y1이 장애물의 y2와 y1사이에 있을 때, User의 x2가 장애물의 x1과 같은 값일 때 닿는다. y축이 장애물과 닿았을 때에는 User의 x1이 장애물의 x1과 x2사이에 있거나, User의 x2가 장애물의 x1과 x2의 사이에 있을 때, User의 y1이 장애물의 y2에 부딪히거나, User의 y2가 장애물의 y1에 부딪히면 닿은 것으로 간주한다. 닿는 순간 게임을 멈춘다.  • 게임이 끝나서 현재점수를 저장하는 부분  SDRAM Base에 저장을 하고, 최고기록은 그 값들 중 가장 큰 값들을 display할 수 있도록 한다. Clear KEY를 누르면 interrupt가 발생하고, KEY\_ISR에서 SDRAM의 값들이 다 초기화될 수 있도록 C언어로 작성한다. 만약 현재 점수가 지금까지 최고점수보다 높으면 최고점수를 리뉴얼 해준다. 그리고 게임을 빠져나와서 while문 안에서 interrupt를 기다릴 수 있도록 한다.  • 장애물이 왼쪽벽에 닿았는지 판단하는 부분  User가 장애물에 닿지 않아서 계속 왼쪽으로 가서 벽까지 닿은 것이므로, 왼쪽 벽의 x좌표인 0에 장애물의 x1이 도달하게 되면 계속 count하면서 현재점수를 +1해준다. 그리고 닿은 장애물은 화면에서 없애 준다. 그리고 새로운 장애물이 나타나도록 설계한다. 그 부분은 장애물이 예를 들어 3개만 나오게 하려면 장애물을 그리는 변수 x와 y를 3개만 선언해주고, 벽에 닿는 순간 계속 돌아가면서 새롭게 x는 360, y는 랜덤하게 다시 초기화를 해주는 방법으로 설계한다. 장애물이 벽에 닿지 않았으면 다시 x좌표를 - speed PIXEL만큼 뺄셈하여 줌으로써 장애물이 다음 frame에 왼쪽으로 움직일 수 있도록 해준다. | | | | | | | |
| 시험/평가 | 검증 계획 | Altera Monitor Program 15.0은 지금까지 프로그램의 작동 여부를 검증하는 데에 사용했던 검증 방법이다. 이 모니터 프로그램의 메모리 값을 확인하는 방법과 printf문 등을 이용하여 i/o interrupt의 발생 여부를 모니터 프로그램의 terminal 창에서 확인하는 방법을 채택하여 각 프로그램이 정상적으로 작동하고 있는지 확인할 수 있다.  • KEY interrupt check  KEY의 ienable과 status register의 PIE bit가 1이면 interrupt가 발생할 수 있는 여건이 제공된다. KEY1을 제외한 KEY들 중 어느 하나라도 눌리면 interrupt가 발생하게 되는데, 이 때 interrupt가 올바르게 발생하였는지 확인할 수 있는 방법은 KEY가 눌렸을 경우에 실행될 프로그램인 KEY\_ISR에 printf문을 넣어주는 것이다. 이렇게 함으로써, exception handler를 거치는 과정에서 interrupt가 올바르게 인식되어 KEY\_ISR이 실행되면 KEY\_ISR이 실행되었음을 알려주는 특정한 문장, 이를테면 "KEY0" 등을 terminal 창에 출력한다. KEY\_ISR 프로그램을 빠져나가기 전 " printf ("KEY0\n"); " 과 같은 문장을 실행함으로써 user가 interrupt 발생 여부를 가시적으로 검증할 수 있다. 프로그램이 정상적으로 작동하지 않을 때 어떤 부분에 오류가 있는지 정확히 파악하려면 이러한 간단한 과정이 오류 검증에 매우 용이하다.  • Audio check  우리 조가 만드는 video game은 user의 목소리를 input으로 받으므로, 이 input이 DE1-SoC의 Audio Port에 잘 입력되고 있는지 확인할 필요가 있다. 이를 위해 mic 이외에도 line out port에 스피커를 연결하여 sampling 된 음성을 실시간으로 출력하도록 한다. 들어오는 Audio 신호를 sampling한 값들을 저장하기 위해, FPGA칩 내부에 있는 SDRAM\_BASE에 동적 메모리 할당으로 메모리를 선언해주고, sampling할 때마다 memory address를 1씩 증가시켜서 저장한다. 결과적으로 mic로 입력되는 input은 연속적으로 memory에 저장된다. 이는 monitor program의 memory 창에 표시된 데이터 값을 확인함으로써 데이터가 메모리에 잘 저장되고 있는지 검증할 수 있다. 그리고 변수를 선언해주어 일정 간격만큼 우리가 임의로 sampling하여 그 사이의 진폭의 최대값을 변수에 넣어준다. 예를 들면 MIC IN에 계속 data가 들어오면 0.1초마다 audio를 sampling하여 그 소리 파형에서 가장 높은 값을 변수에 넣는 것이다.    결과적으로 그림처럼 소리의 크기가 크면 변수의 크기가 커지고, 소리가 작으면 변수가 작아지는데, 이 때, 음이 높아지는 것은 변수의 크기와 전혀 상관이 없다. 검증은 변수의 변화를 둔감하게 하여 같은 사람이 큰 변화를 주지 않으면 변수가 변하지 않도록 설정한 후, 같은 음을 내면서 다른 소리 크기를 냈을 때 변수 값의 변화와, 다른 음을 내면서 같은 소리의 크기를 냈을 때 변수 값의 변화를 printf()를 통해 나타내면서 확인한다.  • Fever mode check(TIMER interrupt & Counter)  일정 시간 동안은 장애물을 피하지 않아도 점수가 증가하며 게임을 계속 진행할 수 있는 'fever mode'는 미리 설정한 시간 동안만 작동시킬 것이므로, 남은 시간 및 진행 시간을 세어 LEDR에 display하기 위해서는 timer interrupt가 필요하다. Fever mode는 5초동안 진행할 것이므로, timer가 5초를 세면 timer interrupt가 발생하고 따라서 TO bit가 1이 된다. TO bit가 1이면 fever mode를 종료하도록 한다. 이 때 쓰이는 timer interrupt는 printf와 LEDR로 검증할 수 있다. Timer interrupt가 발생하여 timer\_ISR이 올바르게 실행이 되면 terminal 창에 반복적으로 timer interrupt를 검증하는 문장이 출력되도록 한다.  •VGA  모니터로 연결해서 가시적으로 바로 확인할 수 있다. 화면을 재생시켜서 가로선으로 이미지 잘림 현상(Tearing)이 나타나면 V-Sync로 문제를 해결하여 주고, 화면 깜빡거림 현상(Flickering)은 Back Buffer를 이용하여 Double Buffuring Tech를 이용하여 해결하여 주도록 한다. 모든 것을 검증하였다면 모니터에서 display가 해당게임의 모든 event를 만족하는지 가시적으로 검증한다.  •최고점수 & 현재점수 & 난이도 Level  Switch를 통해 Level을 조정한 후 printf를 통해서, 현재점수는 1이 증가할 때마다 printf로 monitor program에 출력하여 준다. 최고점수는 게임이 끝나서 현재 점수가 memory에 저장된 후, memory에서 저장한 후 값을 확인하여 가장 큰 값을 printf()해주고, 이 모든 값들은 HEX5부터 HEX0까지 알맞은 bitcode를 이용하여 제대로 display되는지 확인하여 검증한다. | | | | | | | |
| 일정 | | 설계 내용 및 검증 방법에 언급된 전체 진행 과정의 일 정을 자세히 기술한다. 0.5페이지 이상 작성. 아래와 같이 각 주 별로 예상되는 산출물을 정확하고 구체적으로 명시한다.   |  |  |  | | --- | --- | --- | | 주차 | 진행 계획 | 예상 산출물 및 Demo 내용 | | 1 | fever모드를 하기 위해 KEY[0]를 누르면 KEY interrupt를 받아와서 TIMER가 시작하게 된다. TIMER가 일정시간이 지나면 interrupt가 발생하고 그 인터럽트에서는 TIMER를 끌 수 있도록 한다. TIMER가 종료되면 fever모드를 종료한다. fever모드가 얼마나 남았는지는 LEDR[9:0]을 통하여 표현해주도록 한다.  HEX5, HEX4에는 최고기록을 display한다. 지금까지의 기록을 clear하는 KEY[2]를 interrupt로 받으면 지금까지의 기록이 저장되어 있는 memory가 clear되고, HEX5, HEX4로는 0를 display한다. play중 장애물에 닿아서 게임이 끝나는 순간 현재값이 기록에 남아야 하므로, 장애물에 닿는 event를 KEY[3]이 눌리는 interrupt라고 가정하고 KEY[3]이 눌리면 현재 스코어를 메모리에 기록한다. 만약 현재 스코어가 최고기록이면 HEX5, HEX4에 현재의 값을 display하도록 한다.  SW를 통해 level을 설정할 수 있고, 그 level은 HEX1, HEX0를 통해 display되도록 한다. | HEX5, HEX4에는 지금까지의 최고 기록을 display한다. 만약 clear하는 KEY[2]가 눌리면 HEX5, HEX4에 0을 display하도록 한다.  HEX3, HEX2에는 현재 점수를 display한다. 이 때, 현재 점수는 KEY[3]이 눌리면 증가하도록 한다.  HEX1, HEX0에는 현재의 스위치 값을 읽어와서 level을 표시하도록 한다. 실제로, SW값으로 mode를 설정할 수 있도록 한다. fever모드를 동작시키기 위해서 KEY[0]가 눌리면 LEDR[9:0]이 하나씩 감소하면서 fever모드가 얼마나 남았는지 display해 줄 수 있도록 한다. | | 2 | MIC Input을 통해 소리를 입력하고, 그 값의 진폭의 최대값을 일정 주기마다 sampling하여 그 최대값을 구할 수 있도록 한다. 그 최대값이 크면 소리가 크다는 의미이므로 user가 빠르게 움직일 수 있도록 설정하고, 최대값이 작으면 소리가 작다는 의미이므로 user가 천천히 움직이거나 아래로 떨어지는 기능을 구현해야 한다. 구현하기 위해 그 parameter값을 설정해야 하는데, 어느정도 소리를 크게 내야 진폭의 digital화 된 값이 어느정도 나오는지 확인하기 위해 값들을 메모리에 저장한다. 이후 메모리 값들을 읽어서 적절한 상수로 나눠준 후 그 값을 user character의 위로 올라가는 속도로 설정한다. | 진폭의 최대값을 갖고와서 적절한 parameter를 설정해 줄 수 있도록 한다. 녹음기능을 통해 memory에 값들을 저장하고, 저장된 memory 값을 monitor program을 통해서 확인한 후 적절한 값을 나눠준다. parameter라는 변수를 선언하여 소리의 크기를 판단하는 척도를 parameter에 저장한다. 결과값을 monitor프로그램에 printf를 통해 출력할 수 있도록 한다. | | 3 | VGA를 이용하여 장애물이 랜덤한 y값을 갖고 일정한 주기마다 화면 맨 오른쪽부터 생성된다. 장애물은 생성된 위치에서 왼쪽으로 가고, 그렇게 해서 왼쪽화면 끝까지 장애물이 도달하면 현재 점수를 1 증가시켜서 HEX3, HEX2에 display시킨다.  SW를 통해서 현재의 level을 설정하고, level이 높아질수록 장애물의 속도가 빨라지도록 설계한다.  오른쪽 화면 끝에서 랜덤하게 낮은 확률로 fever 모드를 쓸 수 있는 아이템이 생성된다. 아이템을 먹으면 fever모드인 KEY[0]를 눌렀을 때 fever모드가 되고, 만약 아이템을 먹지 않았는데 KEY[0]를 누르면 fever mode가 동작하지 않도록 설계한다.  User character는 소리의 크기에 따라 움직이도록 한다. 2주차에서 확인한 parameter의 값에 관한 식으로 user character의 속도를 결정한다. parameter가 크면 빠르게 위로 올라가고, 소리가 작으면 천천히 올라가거나 떨어지도록 설계한다.  만약 모니터에서 user character가 장애물에 닿으면 게임은 끝나고, 현재 점수가 메모리에 저장된다. 그 값이 최고 점수라면 HEX5, HEX4를 display해준다.  Clear는 KEY[2]로 설정해주고 동작은 1주차에 했던 것처럼 동작하도록 한다. | 1주차에서 한 내용을 조금 수정하여 장애물에 닿으면 게임이 끝나는 것으로 수정한다.  fever모드는 항상 쓸 수 있는 것이 아니라 아이템을 먹어서 갖고 있을 때에만 동작할 수 있다. Fever 모드를 사용하면 소지하고 있는 아이템의 수는 1 감소한다.  SW의 값에 따라 장애물 속도가 달라진다.  장애물이 끝까지 가면 현재 점수가 1 증가한다.  게임이 끝났을 때 현재점수가 최고점수보다 높으면 최고점수를 현재점수로 바꿔준다.  소리가 크면 빠르게 User character가 움직이고, 작으면 천천히 움직이거나 떨어진다. | | | | | | | | |
| 역할 분담 | | • 오승현 : mic에 소리가 입력되었을 때 이를 Audio CODEC을 통해 소리를 처리하고 스피커로 출력하기까지의 전체적인 procedure을 이해하고 이에 맞는 코드를 작성하는 일을 총괄한다. Audio data를 입력 받아 메모리에 저장하는 프로그램을 비롯하여 Audio 신호를 활용한 프로그램들을 구상하고 이에 맞는 알고리즘을 구현하는 과정을 구체화한다. Input으로 들어온 audio data의 음량을 수치화 하여 이에 따른 user 개체가 움직일 상대적인 수직거리를 결정하는 알고리즘을 만들고 이에 맞는 코드를 작성한다. User가 fever mode를 가능하게 하는 아이템을 획득하였을 때 현재 가지고 있는 아이템의 개수를 저장해 두고, KEY0가 눌릴 때마다 1씩 감소시키는 부분을 담당한다. 또한, 지금까지 플레이 한 round들의 점수를 저장하여 이들을 서로 비교하고, 최고 점수를 구하는 파트를 구현한다. 지금까지의 모든 점수들의 기록을 메모리에 저장한 후, 값들을 읽어와서 최고점수를 display하는 파트를 담당한다.  • 이지민 : Fever mode를 가동할 수 있는 아이템을 user가 획득하여 이를 사용하였을 때 Fever mode의 유지 시간 또한 timer interrupt를 이용한 것이므로, TO bit를 이용하여 유지 시간동안 fever mode를 작동하는 부분을 구현한다. Switch / Key / Timer 등의 interrupt를 처리하는 각 i/o들의 ISR과 exception handler 등의 알고리즘을 C language로 코드를 작성하고, 카운터 증감에 따른 HEX display 및 LEDR display를 담당하여 전체적인 게임의 진행 상황을 알리는 디스플레이를 도맡아 한다. user 개체와 장애물 및 아이템 등을 VGA에 표현하고 user가 설정한 난이도에 따라 장애물의 접근 속도 등을 조절하는 알고리즘을 작성한다. 장애물이 왼쪽 벽면에 닿았을 때 이를 인식하고 점수를 증가시킬 수 있도록 하는 부분을 작성한다 | | | | | | | |