논문 2008-6-11

모바일 원-버튼 게임의 구현

Implementation of a Mobile One-Button Game

오선진*, 최세영*, 이영대**

Sun-Jin Oh*, Se-Young Choi*, Young-Dae Lee**

요 약 최근 무선 인터넷 기술의 발전에 힘입어 가장 큰 이슈는 모바일 컴퓨팅 환경에서의 온라인 게임의 설계이다. 현재 이동 단말용 게임들은 낮은 성능의 프로세서, 작은 메모리 공간과 짧은 배터리 파워의 제한 등으로 온라인 게임 구현에 많은 제약이 따른다. 따라서 최근 모바일 게임은 단순하면서 조작하기 쉬운 원-버튼게임이 큰 인기를 얻고 있다. 본 논문에서는 이러한 경향에 맞춰 무선 모바일 컴퓨팅 환경에서의 휴대전화를 이용한 모바일 온라인 게임으로 우리나라 전통 놀이인 제기차기 원-버튼 게임을 설계하고 구현하였다.

Abstract The big issue according to the advance of wireless internet is the design of on-line game under mobile computing environment. The current mobile games have constraints in implementation of on-line game due to the low performance processor. small memory size and short battery time. Therefore, nowadays one-button games which are simple and easy to manipulate, have attracted great interests. In this paper, we designed and implemented the trandition Jeki-chaki game, which is a kind of one-line game, using a mobile handset.

Key Words : 모바일 컴퓨팅, 원-버튼 게임, 무선 인터넷

I. 서 론

무선 인터넷의 발전과 더불어 모바일 컴퓨팅 환경에서 자유롭게 이동하면서 컴퓨팅 할 수 있는 다양한 응용들이 절실히 요구되고 있다. 대표적인 모바일 장비로 휴대전화와 PDA가 널리 대중화를 이루고 있으며, 최근 마니아들을 중심으로 정보 검색이나 동영상, 모바일 게임 등이 여가활동을 하는데 중요한 자리를 차지하게 되었다. 최근 무선 인터넷 사용자들 사이에서 가장 관심의 대상은 단연 모바일 게임이다. 하지만 이러한 모바일 게임들을 운용하는 인프라나 단말장치들은 많은 제약이 따른다. 무선 기반의 인프라는 제한된 대역폭 등으로 데이터 전송의 제약을 받으며, 이동 단말장치들은 낮은 성능의 프로세서, 작은 메모리 공간과 상대적으로 짧은 배터리 파워 제한과 데스크톱 PC와는 비교가 될 수 없는 극히 제한적인 입출력 장치를 갖는다. 이에 이러한 모바일 환경에 알맞은 단순하고, 조작이 용이하고 간편하며, 쉽게 게임을 익히고, 간단한 버튼 조작만으로도 용이하게 게임을 할 수 있는, 그리고 시간과 장소에 구애 받지 않는 무선 모바일 게임의 개발이 절실히 요구되고 있다.[1]

본 논문에서는 이러한 최근의 정보 인프라와 더불어, 자유롭게 이동하면서 컴퓨팅 할 수 있는 모바일 컴퓨팅 환경에서의 대표적인 무선 단말기인 휴대전화를 기반으로 하는 무선 모바일 원-버튼 게임을 설계하고 구현하였다. 본 논문에서 설계하고 구현한 무선 모바일 원-버튼 게임은 타이밍 액션 게임 장르에 속하며, 우리나라 대표적인 민속 놀이중의 하나인 제기차기 게임을 구현하였으며, 단순히 휴대전화의 버튼 하나만으로 모든 게임의 제어가 가능하며, 게임 자체가 단순하면서도 중독성을 가질 수 있도록 힘 조절과 타겟 정확도, 그리고 방향에 따라 변화되도록 설계하였다.

*정회원, 세명대학교 정보통신학부
**정회원, 세명대학교 정보통신학부
접수일자 2008.10.22, 수정완료 2008.12.3

본 논문의 구성은 다음과 같다. 2장에서는 휴대전화 기반 무선 모바일 원-버튼 게임을 위한 시스템 모델을 살펴보고, 3장에서는 본 논문에서 설계하고 구현한 무선 모바일 원-버튼 게임인 제기차기 게임의 구조와 알고리즘들을 자세히 서술하였으며, 4장에서는 본 논문에서 구현한 모바일 환경에서의 제기차기 게임의 구현 및 실행 결과 등을 소개하였다. 그리고 마지막으로 5장에서 향후 연구 내용을 기술하고 결론을 맺는다.

II. 시스템 모델

무선 모바일 컴퓨팅 환경에서의 단말장치의 프로세스 처리 능력은 데스크 탑 컴퓨터에 비해 현저하게 떨어지고, 저장 장치 용량 역시도 상대적으로 매우 작다. 또한 원활한 이동성을 위해 단말 장치에 대한 전원 공급은 주로 배터리를 사용하기 때문에 휴대 단말의 전원 공급에 대한 많은 제약이 따르게 되며, 무선의 낮은 대역폭을 이용하여 데이터를 송수신해야 하므로 데이터 전송과 품질에 대한 신뢰성이 떨어진다. 따라서 이러한 무선기반 모바일 컴퓨팅 환경의 비효율성을 고려하여 무선 환경에서의 온라인 모바일 게임의 형태는 지극히 제한적인 성능을 가진 단말장치에서 지극히 제한적인 입출력 장치를 통해 최소의 데이터 교환을 하게 하면서, 게임을 운영할 수 있는 형태가 요구된다. 이러한 경향으로 인해 휴대전화를 기반으로 하는 모바일 온라인 게임의 형태는 단순하고, 조작이 용이하고 간편하며, 쉽게 게임을 익히고, 간단한 버튼 조작만으로도 용이하게 게임을 할 수 있는, 시간과 장소에 구애 받지 않는 모바일 온라인 게임들이 각광을 받고 있는 실정이다.[2, 3] 다음의 표 1은 본 논문에서 구현한 무선 모바일 컴퓨팅 환경에서의 원-버튼 게임을 위한 시스템 모델을 보여준다.

표 1. 시스템 모델
Table 1. System Model

모바일 원-버튼 제기차기 게임 시스템 모델	
게임개발 PC 사양	PC (Pentium 3급 이상)
운영체제	Windows 98 이상 (XP Professional)
개발 도구	Aroma wipi-Clet v. 1.1
개발 언어	MS Visual Studio C++ 6.0 이상
모바일 플랫폼 사양	wipi가 탑재된 Mobile Phone (SKT, KTF, LGT) 또는 에뮬레이터

제기차기 게임은 휴대 전화를 기반으로 하는 온라인 게임의 개발인 만큼 휴대전화 서비스를 제공하는 통신사의 서비스 환경을 고려해야 한다. 일반적으로, SKT, KTF, LGT 등 휴대전화 서비스를 하는 각 통신사마다 모두 고유의 플랫폼을 가지고 있으나, Wipi의 경우는 표준 규격이라 어느 통신사 플랫폼에서도 사용이 가능하다. 본 논문에서 게임 개발을 위해 채택한 모바일 플랫폼으로는 아로마 위피 1.1 버전을 사용하였고, 게임 개발을 위해 사용한 프로그래밍 언어는 마이크로소프트사의 비주얼 스튜디오 c++를 사용하였다. Wipi 플랫폼에서 사용 가능한 프로그래밍 언어는 c와 java 등이 있다. 한편 sk-vm, midp, wipi-jlet는 java를 기반으로 동일한 소스로 포팅이 가능하며, brew, gvm, gnex 등은 c를 기반으로 동일한 소스로 포팅이 가능하다.[4,5] 아로마 wipi 플랫폼에서는 jlet와 clet 모두를 사용할 수 있지만 서로 사용한 프로그래밍 언어에 따라 명확히 jlet인지 clet인지 구분해서 사용해야 된다. 즉 완성된 게임의 실행을 위한 배치 파일을 만들 경우 java를 사용했을 경우에는 jlet를, c를 사용했다면 clet를 써야한다. 왜냐하면 jlet를 clet로 포팅하면 에러가 발생하고 이를 해결하기 위해 소스 수정을 하는 것은 매우 복잡하고 시간이 오래 걸리기 때문이다.[6]

III. 원-버튼 게임의 설계

본 논문에서 설계하고 구현한 모바일 원-버튼 게임은 "제기차기" 게임으로 모바일 플랫폼인 WIPI를 기반으로 하는 원-버튼 타이밍 액션 장르의 게임이다. 우리나라 고유의 민속놀이인 제기차기를 핸드폰에서 즐길 수 있는 모바일 게임으로 실제 제기 차기를 누가 오랫동안 지속시킬 수 있는 지 그리고 또 제기를 얼마나 높이 찰 수 있는 지를 경쟁하는 게임이다. 휴대전화를 기반으로 하는 모바일 게임이 주로 이동 중이나 잠깐 짬이 날 때 하게 되는 게임이므로, 단순하면서 중독성이 있다는 것이 '원-버튼 게임'의 장점이라 할 수 있다. 전통의 소재에 맞게 게임 내의 그래픽은 한지에 붓으로 그린듯 한 전통적인 느낌의 화면을 구현하였으며, 게임의 캐릭터 역시 전통적인 분위기가 풍기도록 붓으로 한번 그린듯 한 모습을 가진 '일획'이란 이름의 캐릭터를 사용한다. 또한 게임내의 캐릭터가 이동할 때 잔상효과를 연출하여 지금까지의

모바일 게임에서는 볼 수 없었던 새로운 느낌과 캐릭터의 움직임을 보다 역동적으로 표현하는 그래픽을 보여준다. 이러한 잔상효과는 캐릭터의 움직임에 흔적이 남아 서서히 없어지는 듯하게 표현하는 것으로, 주로 너무 빠르게 움직이거나, 신비한 느낌을 표현할 때 쓰인다.

이러한 잔상효과를 표현하기 위해 비트 맵의 투명도를 나타내는 알파 값을 이용하였다. 즉, 게임의 배경과 잔상효과를 낼 캐릭터를 적당한 알파 값으로 출력하여, 플리핑 이전의 이미지를 완전히 덮어버리지 않고 알파 값으로 적당히 겹치게 해서 잔상처럼 보여 지게 하는 방법이다. 우선 먼저 알파 값을 적당히 맞추고, 배경과 캐릭터를 출력하여 알파 효과를 없앤 다음, 흰 배경에 알파 효과를 주어 출력하게 되면 약간 색상이 연해지기만 할 뿐 크게 달라져 보이지 않는다. 이때 배경을 포함한 전체 화면을 플리핑한다. 이렇듯 잔성 효과의 원리는 간단하다. 이전 이미지가 출력되어 있는 LCD화면에 알파 값을 가진 새로운 배경이 출력되면서, 이전 이미지의 캐릭터가 잔상처럼 뿌옇게 되고, 이때 배경은 새로운 배경과 겹치면서 아무런 변화가 없어 보인다. 그 위에 새로운 캐릭터가 출력되면서 잔상과 함께 그려지는 모습이 되는 것이다.

잔상 효과와 더불어 게임의 그래픽 처리의 향상을 위해 더블 버퍼링 기법을 도입한다. 더블 버퍼링 기법은 그래픽 출력장치와 메모리 속도 간의 병목현상으로 생기는 속도 저하를 피해야하는 게임에서 두 개 혹은 그 이상의 비디오 메모리를 확보하는 기법을 말한다. 화면에 표현하는 것은 표현 로직, 그래픽 카드, 그리고, 모니터가 필요하다. 이러한 세 가지의 구성요소는 각각 독립적으로 실행되며, 표현하기 위한 데이터를 통신하게 된다. 이러한 요소들이 독립적인 실행에 따른 표현의 어색함이 발생하게 되고, 모니터 화면이 깜빡이는 현상 등을 만들게 된다. 이러한 문제를 해결하는 것이 더블버퍼링 기법이다. 즉, 메모리 버퍼를 여러 개 나눠서 그림을 그리고, 그 그림 전체를 한 번에 비디오 카드 영역에 옮기는 기법이다. 본 논문에서 사용한 더블버퍼링 기법은 시작점과 끝점에 이동하는 것을 모두 버퍼에 그림을 그리고 끝점에 가까울수록 알파 값을 연하게 시작점에 가까울수록 알파 값을 진하게 넣어서 그래픽 처리 속도를 높이고 좀 더 잔상 효과를 생생하게 하였다. 즉, 더블버퍼링을 사용함으로써 메모리에 접근하는 것이 빠르기 때문에 그림을 그리는 데에는 시간이 적게 들고, 메모리 영역의 그림을 비디오 영역으로 보내는 BitBlt 함수가 일괄적으로 처리되기 때문에 BitBlt함수의 시간이 LineTo 함수와 소요시간이 크게 틀리지 않게 된다.

IV. 구현 결과

본 논문에서 구현한 모바일 원-버튼 게임인 "제기 차기" 게임의 구성은 다음과 같다. 우선 기본적으로 4개의 헤더파일과 3개의 소스파일로 구성되어있으며, wipiHeader.h 파일은 아로마 위피에서 제공해 주는 헤더파일이다. Define.h는 게임의 처음 시작부터 해서 사용되는 모든 그림들의 경로를 지정하고 설정해주는 부분이다. 배경화면과 캐릭터 등등 사용되는 모든 것은 여기서 지정을 해주어야 메인 파일이나 C파일에서 사용이 가능하다. Function.h는 게임 프로그램에서 사용할 함수들을 선언하는 함수선언 헤더 부분이다. Variable.h는 게임에서 사용되는 이미지, 캐릭터 , 게임 메인의 구조체를 선언해주는 부분이다. FrameWork.c는 게임의 실행 시 일어나는 특정 이벤트 조건에 따라 게임을 시작하거나 종료해준다. Jeki.c는 실질적인 게임모드 '제기차기'에 관한 함수들을 포함하는 파일이고, MinsokMain.c 파일은 제기차기 게임 실행의 메인 부분이다.

```
void usCreateImage(char* filename, sImgData* img)
{
    M_Byte buff[256];  //임시버퍼 생성
    M_Uint32 mBufID; //버퍼ID
    M_Int32 size;     //버퍼사이즈
    M_Int32 ret;      //MC_grpCreateImage 리턴값
    MC_knlSprintk(buff, filename);  //버퍼에 파일명저장

    img->resID = MC_knlGetResourceID(buff, &size); //파일명으로 size
    mBufID = MC_knlCalloc(size);  //해당 이미지 size만큼 동적할당
    MC_knlGetResource(img->resID, (void*)mBufID, size);
  //할당한 메모리 주소값을 mBufID에 resID에 복사
    ret=MC_grpCreateImage(&img->img, mBufID, 0, size); //이미지생성
    img->loaded = 1;  //img구조체의 이미지로드유무 Flag 설정
    MC_knlPrintk("Load image > ret: %d, buf id: %d, size: %d\n", ret,
    mBufID, size); //디버그용 콘솔 출력
    img->resID = mBufID;  //img구조체 redID에 메모리주소값 설정
}
```

그림 1. FrameWork용 이미지 생성 알고리즘
Fig.1. Image generation algorithm for framework

그림 1은 Wipi 기본함수 및 자주 사용하는 패턴을 함수로 제작한 Framework 용 함수를 포함하는 이미지 생성 알고리즘으로 이 부분에서 게임의 실행 시 일어나는 특정 이벤트 조건에 따라 게임을 종료하거나 시작하게 해준다. 그림 2는 Jeki.c 파일 내의 실질적인 제기차기 게

임과 관련된 함수들을 정의해 놓은 부분으로 여기서 캐릭터 이미지의 잔상효과를 다루는 알고리즘을 보여준다.

```
int usJekiRushSprite()
{
        //잔상효과
        if(pGameMain->m_imgHead.y == 24 )
       //같은 것을 표현하는 약간 다른 스프라이트를 이용
        {
            pGameMain->m_imgHead.y = 0;
            pGameMain->m_imgBody.y = 0;
            pGameMain->m_imgLeg.y = 0;
        }
        else   //서로 반복하여 출력하며 잔상효과를 표현.
        {
            pGameMain->m_imgHead.y = 24;
            pGameMain->m_imgBody.y = 40;
            pGameMain->m_imgLeg.y = 40;
        }

        //Tic Toc
        if(pGameMain->m_RushTic > pGameMain->m_RushToc)
        {
            pGameMain->m_RushTic = 0;
            pGameMain->m_sRush.RushStateCnt++;
        }else
        {
            pGameMain->m_RushTic++;
            return 0;
        }
```

그림 2. 제기차기 이미지 잔상효과 알고리즘
Fig. 2. Afterimage effect algorithm in Jeki-chaki game

```
void usInitClet()
{
    appMemID = MC_knlCalloc(sizeof(CletApp));
    //게임에 사용되는 변수들을 포함한 Clet 구조체를 선언
    //MC_knlCalloc으로 동적할당함으로써 객체화
    pGameMain= (CletApp*)MC_GETDPTR(appMemID);
    pGameMain->bpl = MC_grpGetScreenFrameBuffer(0);
    MC_grpInitContext(&pGameMain->cxt); //그래픽 콘텍스트 초기화

        {중략}

    //이미지 로드
    usCreateImage(IMG_BG,&pGameMain->m_imgBg);      //배경1
    usCreateImage(IMG_BG2,&pGameMain->m_imgBg2);    //배경2
    usCreateImage(IMG_TEXT,&pGameMain->m_imgText); //배경2
    usCreateImage(IMG_HEAD,&pGameMain->m_imgHead);
    //제기차기 캐릭터 머리 스프라이트
    usCreateImage(IMG_BODY,&pGameMain->m_imgBody);
    //제기차기 캐릭터 몸 스프라이트
    usCreateImage(IMG_LEG,&pGameMain->m_imgLeg);
    //제기차기 캐릭터 다리 스프라이트
    usCreateImage(IMG_DANPOONG,&pGameMain->m_imgDanpoong);
    //단풍배경
    usCreateImage(IMG_JEKI,&pGameMain->m_Jeki.m_imgJeki);   //제기
    usCreateImage(IMG_GAUGE,&pGameMain->m_imgGauge);    //게이지
    usCreateImage(IMG_MAIN,&pGameMain->m_imgMain);      //메인화면
    usCreateImage(IMG_LOGO,&pGameMain->m_imgLogo);      //로고

        {중략}

}
```

그림 3. 제기차기 게임실행 메인 알고리즘
Fig. 3. Main algorithm of Jeki-chaki game execution

그림 3은 제기차기 게임 실행 메인 알고리즘을 보여준다. 게임에 사용되는 변수들을 포함하는 구조체를 선언하고 그래픽 콘텍스트를 초기화 한 후 배경 이미지부터 차례로 캐릭터들의 이미지를 로드한 후 게임이 시작된다.

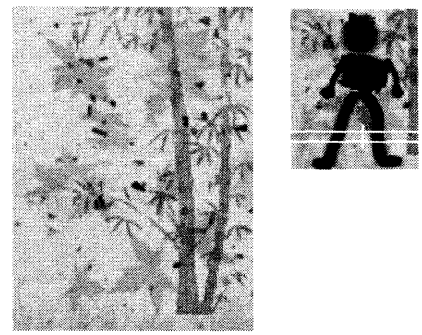

그림 4. 제기차기 게임 배경과 캐릭터 '일획'
Fig. 4. Background of Jeki-chaki game and a character

그림 4는 본 논문에서 구현한 '제기차기' 게임에서 사용한 동양화 느낌의 배경화면과 실제 제기차기를 하는 캐릭터 '일획'의 모습을 보여준다. 더블버퍼링 기법과 알파 값을 조정한 잔상효과를 도입한 모습을 보여준다.

게임의 목표는 제기를 높이, 그리고 오랫동안 차는 것이다. 높이 찬 길이 1cm마다 1점, 찬 횟수 1회마다 10점으로 점수가 부여된다. 제기를 차는 방법은 간단하다. OK버튼을 누르고 있으면 게이지 바가 차는데, 누르고 있던 버튼을 떼면 찬 게이지의 파워만큼 제기를 차게 된다. 게이지가 차면 제기를 차는 힘에 세져, 제기를 높이 찰 수는 있지만, 정확히 타이밍을 맞추지 못하면 제기가 좌, 우로 날아가 버릴 가능성이 높다. 너무 멀리 가면 캐릭터가 쫓아가지 못해 제기가 땅에 떨어지고 이때 게임이 끝난다. 반대로 게이지를 채우지 않고 차게 되면, 제기가 높이 올라가지 않아, 다음 찰 때 타이밍을 놓치기 쉽다. 정중앙의 힘으로 차야 안정적인 힘으로 오랫동안 찰 수 있다. 게이지보다 더 중요한 것은 버튼을 놓는 순간, 즉 제기를 차게 되는 타이밍이다. 너무 빨리 차거나 너무 늦게 차면 헛발질을 해 제기를 땅에 떨어뜨리게 되어 게임이 끝나게 된다. 제기를 차게 되었다 하더라도, '스트라이크존' 정중앙에서 차지 않았다면, 제기는 다른 방향으로 가서 다음 번 찰 때 타이밍을 잡기 어렵게 된다.

보통은 오른발로 제기를 차지만, OK버튼을 연속으로 두 번 누르면 제기 왼발차기가 된다. 키를 두 번을 눌러야 하므로 입력은 불편하지만, 왼발로 찰 타이밍에 왼발로 차게 될 경우, 보다 안정적으로 제기를 차게 된다. 그

림 5는 제기차기 게임을 실행해서 제기를 스트라이크 존에 놓고 제기차기를 하는 화면과 제기차기 게임이 끝나고 나서, 점수가 정산되어 게임 성적과 함께 종료를 알리는 화면이다.

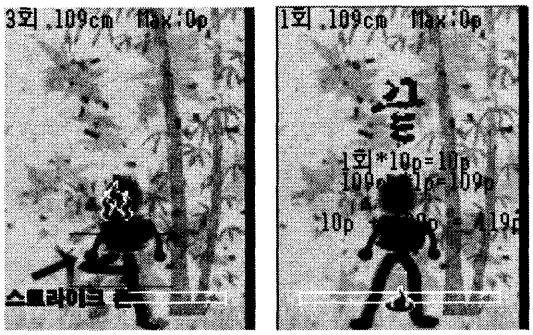

그림 5. 제기차기 게임 실행과 게임 종료 화면
Fig. 5. Execution and end of Jeki-chaki game

V. 결 론

본 논문에서는 시간과 장소에 구애 없이 자유롭게 이동하면서 컴퓨팅 할 수 있는 모바일 컴퓨팅 환경에서의 대표적인 무선 단말기인 휴대전화를 기반으로 하는 무선 모바일 원-버튼 게임을 설계하고 구현하였다. 본 논문에서 설계하고 구현한 모바일 온라인 원-버튼 게임은 타이밍 액션 게임 장르로 우리나라 대표적인 민속 놀이중의 하나인 제기차기 게임을 구현하였으며, 단순히 휴대전화의 버튼 하나만으로 모든 게임 제어가 가능하고, 단순하지만 중독성을 가지도록 다양한 변화를 넣어 설계하였다. 무선 단말기가 갖는 제한된 메모리나 대역 등 자원의 제약을 위해 게임을 단순화 하고 더블버퍼링 기법과 알파값을 조정한 잔상효과 등을 구현하여 게임의 실행을 현실감 있게 구현하였다. 향후 연구 계획은 온라인상에서 제기차기 게임이 싱글 플레이 뿐만 아니라 멀티 플레이가 가능하도록 하는 것이다.

참 고 문 헌

[1] John W. Fisher II, "Methods and Considerations in Online Game Design," Master's Thesis at Michigan State University, East Lansing, MI USA, 2003.
[2] Jim Adams, Programming Role Playing Games, Course Technology PTR, pp. 974, 2002.
[3] Charles Petzold, Programming Windows, 5th ed. Microsoft Press, pp. 1536, 2005.
[4] 오선진, 임베디드시스템 소프트웨어 개발방법론, 한올출판사, pp. 461, 2007.
[5] 배석희, 한상홍, 전영준, Wipi 모바일 프로그래밍, 대림출판사, pp. 557, 2005.
[6] 이상부, 모바일 프로그래밍을 위한 Wipi, 도서출판 연학사, pp. 410, 2005.

저자 소개

오 선 진(정회원) 제 6 권 제 2 호

이 영 대(종신회원) 제 7권 제 2호

최 세 영(정회원)

세명대학교 정보통신학과