# 모바일 환경에서 GVM 콘텐츠의 BREW 콘텐츠 자동 변환 도구 개발

조준호* · 홍철운** · 이양선***

## 1. 서 론

국내외 모바일 게임 산업의 성장과, 모바일 게임 플랫폼의 다변화 추세는 개발사들로 하여금 단일 모바일 콘텐츠의 타 플랫폼 이식 작업에 대한 필연성을 유발하게 하였으며, '원소스 멀티유즈(OSMU)', '원소스 멀티플랫폼(OSMP)'에 대한 욕구는 갈수록 가중되고 있다. 이러한 가운데에, 콘텐츠의 타 플랫폼 이식 개발을 신속하고 정확하게 진행할 수 있도록 하기위한 여러 가지 노력들이 있어왔으며 이에는 크게 4가지 형태가 존재한다. 첫째는, 플랫폼 자체의 이식이다. GVM 플랫폼은 WIPI 플랫폼의 가상기계(Virtual Machine) 어플리케이션 형태로 작동하여, WIPI 플랫폼에서 GVM으로 개발된 게임 콘텐츠를 직접 구동시키도록 구현되었다. 둘째는, 목적 플랫폼 용 컴파일러이다. 예를 들자면, Java로 개발된 소스를 컴파일 하여, 윈도우즈용 바이너리 파일을 생성하는 기술 등이다. 셋째는, 소스 코드 변환 방식이다. 소스코드를 의미적으로 동등한 목적 플랫폼 언어의 원시파일로 변환하는 방식이다. 넷째는, 멀티플랫폼 엔진이다. 이는 특정 형태의 게임 개발 시에, 여러 플랫폼으로의 적용이 가능한 중간 형태의 API 및 라이브러리를 구축함으로써, 목적 플랫폼에 이식하기 위한 최소한의 노력을 들이기 위한 기술로, 콘솔 및 PC 온라인 게임 등에 사용되어지고 있다.

모바일 게임 분야에서는 현재까지 플랫폼 개발사(신지소프트, 퀄컴 등) 이외에는 이러한 변환 솔루션 구현을 위한 노력이 별도로 연구되어지고 있는 바가 없으며, 특히 일본의 자동변환도구에 있어서는, 일본 플랫폼에 대한 개발 정보와 서비스 경험을 가진 업체들이 부족하여 더욱 진행이 어려워왔다.

현재까지, 자사 및 여러 모바일 게임 개발사들은 일본 현지 플랫폼 기술 정보의 빈곤과, 국내 개발자들의 플랫폼 변환 기술에 대한 이해 부족으로 인하여, 모바일 게임 이식 작업을 위해 많은 시간의 개발 비용과 다양한 시행착오를 겪어 왔다. 이러한 개발 비용의 증가는 신규로 일본 수출을 진행하고자 하는 게임 개발사들 뿐 만 아니라, 기존 일본 수출을 진행하고 있는 개발사에게도 큰 장애 요소로서, 대일본 수출 사업의 활성화를 가로막는 핵심적인 문제점이었다. 그러나 이러한 문제점에 대한 인식에도 불구하고, 국내외적으로 실질적으로 개발비용을 축소시킬 수 있는 대안 마련을 위한 연구 및 개발이 미미해 왔던 것이 사실이다.

이런 이유로 가장 많은 모바일 게임 콘텐츠를 양산해낸 국내 모바일 게임 플랫폼인 GVM으로

* (주)테크론시스템 대표이사
** (주)엠버튼 대표이사
*** 서경대학교 컴퓨터공학과 교수

개발된 게임을 국내 KTF-BREW로 자동 변환해 주는 솔루션 개발을 위해 그동안 진행해 왔던 연구를 토대로, GVM 게임을 일본 KDDI-BREW 게임으로 자동 변환 할 수 있는 도구를 연구 개발하게 되었다.

## 2. 개발방법론과 요구사항 분석

### 2.1 개발방법론

본 논문에서는 GVM2KDDIBREW 자동변환 도구를 개발하기 위하여, 그림 1과 같이 폭포수 모델'(WaterFall Model)을 적용하여 개발하였다. 폭포수 모델은 경험이 풍부한 프로젝트 진행에 있어서 최적화된 모델이므로 본 연구의 개발 방법론으로 적합한 방법론이다.

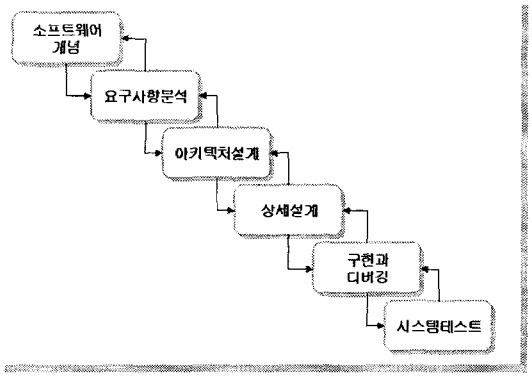



그림 1. 폭포수 모델

### 2.2 시스템 구성도

목표 시스템은 그림 2의 시스템 구성도에서 보는바와 같이, GVM 게임 소스코드 및 이미지, 사운드 데이터를 토대로 하여, BREW 플랫폼에서 동일한 기능을 수행하는 게임에 필요한 소스코드 및 이미지, 사운드, 그리고 추가적인 데이터들을 생성하는 기능을 제공한다. 이를 위해서, 소스, 이미지, 사운드 등 변환 기능과 일본어 출력을 지원하기 위한 기능 및 기타 GVM과 동일 기능을 제공하기 위한 부가적인 기능들이 제공되어져야 한다.

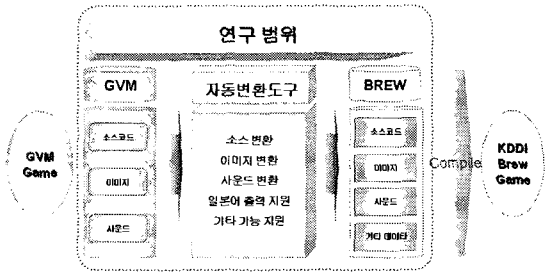



그림 2. 시스템 구성도

### 2.3 요구사항 분석

1) 목표 환경 분석

요구 사항 명세서를 작성하기 위하여, 먼저 소스 플랫폼인 GVM과 목표 플랫폼인 KDDI의 차이점에 대해서 분석하였으며, 현재 GVM2KTFBREW 기술의 응용 발전을 위하여 KTF-BREW와 일본 KDDI-BREW와의 차이점에 대해서도 분석하였다. 또한, 각 플랫폼을 채택하고 있는 이동통신사의 모바일 게임 상용화를 위한 검수 기준의 차이점을 분석하여, 본 연구의 목적인 '상용화 가능한 자동 변환 기능'을 만족할 수 있기 위한 기초 자료를 확보하였다.

GVM과 KDDI-BREW의 차이점을 본 연구의 목적을 부합시키기 위한 기능적 관점에서 분석한 결과는 표 1과 같다.

2) 기능요구사항 분석

최종적으로 개발된 시스템은 GVM 플랫폼의 디스플레이 및 이벤트 환경을 동일한 방법으로 사용할 수 있는 기능과 이미지, 사운드, 진동, 파일 시스템과 관련된 예약어 및 함수들을 BREW에서 동일하게 사용할 수 있도록 제공해야 한다.

본 논문에서 제공하는 GVM2BREW 자동변환

표 1. GVM과 KDDI-BREW 비교

| 분류 | | GVM | KDDI Brew | 해결 방안 |
|---|---|---|---|---|
| 프로그래밍 언어 문법 체계 | | mini C 기반에 문법이지만 GVM에서 지정한 고유 함수만을 사용하여야 한다. image, string, sound등의 자체적인 자료형과 swdata등 시스템에서 미리 예약한 예약어들이 존재한다. | C++을 지원. 하지만 사용가능한 API는 ansi 표준이 아닌 퀄컴에서 만들어 놓은 API를 사용한다. | Brew API를 이용하여 GVM의 예약어와 함수군을 모방할 수 있도록 하여야 한다. |
| 그래픽 | Display 환경 | 0~6 단계의 Gamma를 조절 기능. 현 출력 상황을 저장하는 기능. 그래픽 출력에서 생상 지정은 GVM에서 규정한 0~256 사의 값을 지정하는 방식. | Gamma 조절 기능은 없음. 색상 변경 방법은 R, G, B 값을 조정하는 방식 | GVM의 Gamma 기능과 0~255 사이에 값으로 색상 변경이 가능하도록 하여야 한다. |
| | 백터 | 점, 수직선, 수평선, 임의의 두 점을 끝으로 하는 직선, 사각형, 안쪽이 채워진 사각형, 타원형, 안쪽이 채워진 타원형 | GVM과 동일한 기능을 지니고 있다. | |
| | 이미지 | Variable Depth image(이하 VDI)를 이미지 데이터를 가지고 단순 출력 역상 출력, 색상 변환 출력 | BMP, PNG, 일부 폰에서는 GIF 형식의 이미지 데이터를 출력 할 수 있다. | VDI 이미지 데이터를 Brew에서 출력이 되는 형식으로 변경하여야 한다. |
| | 문자 | 문자 출력 속성인 색상, 크기, 정렬 방식을 조절하는 기능과 설정된 문자 출력 속성에 맞추어 문자를 출력 단말기 독립적인 폰트를 가지고 있다. | 기능은 GVM과 동일하지만 단말기 마다 문자 형태와 크기가 다르다. | GVM과 유사한 형태의 자체 font를 출력는 기능을 구현하여야 한다. |
| 사운드 | | 이미지 데이터처럼 변수 형태로 사운드 데이터를 접근. 사운드 출력과 종료 기능은 함수 호출 한번으로 간편하게 작동시킬 수 있다. | BREW의 리소스 편집기를 이용하여 사운드 자원을 등록, 사운드 출력과 출력 중 중단 기능을 수행하는 동작에서 콜백을 이용한 미세한 제어 | 변수로써 사운드 데이터를 접근할 수 있게 하고,<br>GVM과 동일한 수준의 간편함을 가진 사운드 출력 기능을 지원하도록 하여야 한다. |
| 진동 | | 1sec 단위의 지동을 발생 시키는 기능과 진동을 중단 시키는 기능을 지원하며, 어떤 준비 작업 없이도 간단한 명령문 하나만으로도 동작이 가능하다. | 진동에 관한 동작을 실행시킬 때마다 진동 제어 시스템을 준비 시켜야 한다. 진동 발생 단위가 1/1000sec 단위이다. | GVM 환경과 동일하도록 제반 프로세스를 구축하여야 한다. |
| 저장 | | 최대 64byte 로 이루어진 int 배열을 어떤 준비 작업 없이 쓰기 함수와 읽기 함수 2가지만으로 시스템이 미리 지정한 공간에 쓰고 읽는다. | file 생성, 삭제, 읽기, 쓰기 등이 가능하여 시스템이 허용하는 32bit 만큼 제어가 가능하다. file 제어 시스템을 준비하여야 한다. | GVM 환경과 동일하도록 제반 프로세스를 구축하여야 한다. |
| 네트워크 | | 별다른 부가적 설정 없이도 연결 설정, 연결 재설정, 데이터 송신, 데이터 수신 이벤트 처리, 네트워크 기능 오류 이벤트 처리 현제 네트워크 상태 조회 등이 각각 하나의 함수에 배정 되어 사용자는 간단한 함수 동작 실행만을 함으로써 해당 기능을 동작하도록 지원하고 있다. | 접속을 위하여 네트워크 관리자 생성과 소켓 초기화를 포함한 초기화 과정과 네트워크 송수신 과정 동안에 해당 자원들을 유지 시켜야하며, 데이터 송신을 위한 명령어 set들이 다르고 수신 과정에서 Brew 플랫폼 고유의 수신 이벤트 처리자를 사용하며, 네트워크 기능 오류 상황 이벤트 처리자 또한 Brew 플랫폼 고유의 이벤트 처리자와 시스템 적으로 연결 | GVM 동작과 유사한 형태를 지원하기 위하여 Brew 플랫폼의 네트워크 동작을 사용하여 네트워크 동작을 관리하는 시스템을 구축하여야 한다. |
| 사용자 이벤트 | | 키 이벤트를 처리하는 함수가 존재, 키 이벤트 값은 GVM 내부 규약으로 지정한다. 폴수신과 휴대폰 폴더가 닫혀지므로 필요한 게임이 사용한 자원 반납과 복원에 대한 처리는 시스템이 자체적으로 지원한다. | 키 이벤트를 처리하는 함수가 존재하나 키 이벤트 값은 Brew 플랫폼 고유의 값을 사용. 자원을 반환하고 복원하는 기능은 해당하는 이벤트가 발생하였을 때 이벤트 처리 함수는 동작하지만 구체적인 자원 반환과 복원에 대한 처리는 사용자의 몫으로 남겨놓음. | Brew 플랫폼으로 들어오는 Key 이벤트 값을 변경하여 GVM 규격에 맞춘 Key 이벤트 값으로 바꾸어 주는 구조를 만들고, 자원 관리부를 만들어 생성된 자원 사항들을 기억하고 있다가 기억된 자원들을 반환하고 복원하는 기능을 수행하도록 한다. |

도구 시스템이 제공해야 할 기능은 다음과 같다.

(가) GVM 플랫폼과 동일한 디스플레이 및 그래픽 환경 제공

- 팔레트 환경 제공 : '디스플레이 환경'은 GVM 플랫폼에서 제공하는 256색 팔레트와 동일한 RGB 색상 값을 사용하는 1BYTE 팔레트 환경을 제공해야 한다.

- BackBuffer 제공 : '디스플레이 환경'은, GVM 플랫폼의 SaveLCD(), RestoreLCD() 등의 API를 활용할 수 있도록, 현재 LCD상에 표현된 화면 출력 내용을 BackBuffer에 복사할 수 있고 다시 화면상에 출력할 수 있는 기반 환경을 제공해야 한다.

- 벡터 그래픽 지원 환경 제공 : '디스플레이 환경'은 'GVM 그래픽 API'들이 제공하는 점, 수직선, 수평선, 임의의 두 점을 끝으로 하는 직선, 사각형, 안쪽이 채워진 사각형, 타원형, 안쪽이 채워진 타원형 등을 지정한 색상 값으로 화면에 출력하는 2차원 그리기 함수들을 지원하기 위한 기반 환경을 제공해야 한다.

- VDI 이미지 출력 : '디스플레이 환경'은 GVM에서 사용되는 'VDI(Variable Depth Image) 형식'의 이미지를 출력할 수 있기 위한 기반 환경을 제공해야 한다.

- 문자 출력 지원 : '디스플레이 환경'은 문자열을 GVM과 동일한 형식으로 출력할 수 있기 위한 기반 환경을 제공해야 한다.

(나) GVM 플랫폼과 동일한 이벤트 환경 제공

- 키입력 이벤트 : 사용자가 키를 입력하는 경우 GVM 플랫폼의 KEY_PRESSED에 해당하는 키 이벤트와 동일한 형식 및 이름의 이벤트를 발생시켜야 한다., 또한, 이를 처리할 수 있는 이벤트 핸들러인 EVENT_KEYPRESS()와 동일한 이름의 이벤트 핸들러를 제공해야 한다.

- 타이머 이벤트 : GVM 플랫폼에서 제공하는 3개의 타이머를 동일하게 제공할 수 있어야 하며, 타이머가 호출된 이후 발생되는 타이머 이벤트와 동일한 형식 및 이름의 이벤트를 발생시켜야 한다. 또한, 이벤트 발생 시, 이를 처리하는 EVENT_TIMEOUT()과 동일한 타이머 핸들러를 제공해야 한다.

- 시스템 이벤트 : GVM의 경우 어플리케이션 구동 시, 종료 시, 콜 수신시, 그리고 통화 종료 때에 발생하는 이벤트에 대한 처리는 시스템이 맡아서 처리해 준다. 그럼으로, BREW 플랫폼 검수 환경의 요구 조건을 만족하면서, GVM의 시스템 이벤트 처리와 동일한 기능을 제공하기 위하여, 게임에서 사용되는 자원에 대한 반납, 시스템 객체 해제 및 생성 등의 기능을 제공할 수 있어야 한다.

(다) GVM 플랫폼과 동일한 이미지 사용 환경 제공

GVM 개발 환경에서는 이미지를 사용하기 위해 'ImageMaster' 도구의 출력물인 SBM 파일을 소스에 include하는 방식을 이용한다. SBM파일에는 VDI 형식의 이미지 및 이미지 배열이 선언 및 정의되어 있다. VDI 형식을 이용하여 디스플레이 시, Gamma, Palette변환, 역상 출력 등을 지원하고, 본 이미지 형식에 직접 Binary형식으로 접근 및 수정하는 API를 제공하므로, 본 자동변환 도구에서는 SBM 파일을 기반으로 VDI 이미지 형식을 사용할 수 있는 환경을 제공해야 한다.

(라) GVM 플랫폼과 동일한 사운드 사용 환경 제공

GVM에서는 이미지 데이터처럼 변수 형태로 사운드 데이터를 접근하며, 사운드 출력과 출력 중 중단 기능을 지원하며, 사전 준비 작업 없이도 간단한 API 호출을 통해서 동작되도록 구현되어

있으므로, BREW 환경에서 동일한 방식으로 사운드 입출력을 제어하기 위한 환경을 제공해야 한다.

(마) GVM 플랫폼과 동일한 진동 발생 환경 제공

GVM에서는 1sec 단위의 지동을 발생 시키는 기능과 진동을 중단시키는 간단한 API를 통해 진동 기능을 제어한다. BREW에서도 동일한 환경을 제공해야 한다.

(바) GVM 플랫폼과 동일한 데이터 저장 환경 제공

GVM에서는 사용자 데이터를 저장하기 위해, 사용자가 지정한 정수형 배열과 배열의 길이를 입력으로, 시스템에서 제공하는 Flash Memory에 일괄 저장하는 기능을 지원한다. 또한, 저장된 데이터를 읽어오기 위해, 사용자가 지정한 배열과 읽어올 길이를 받아 일괄 읽기하는 기능을 지원한다. KDDI-BREW 환경에서는 Flash Memory를 제공하지 않으므로 동일한 저장환경을 제공하기 위한 방법이 필요하다.

(사) GVM 플랫폼의 한글과 동일한 규격의 일본어 문자 출력 환경 제공

GVM에서는 한글의 경우 가로 12 Pixel, 세로 12 Pixel의 환경을 제공한다. KDDI-BREW 환경에서는 핸드폰 마다 지원하는 폰트의 크기가 상이하기 때문에, 일본어 출력 시에, GVM 소스 상에서 규정된 문자열의 화면출력과 상이한 문자열 출력 화면을 제공하게 된다. 그러므로, 이러한 환경을 근본적으로 극복하기 위한 문자 출력 환경을 제공해야 한다.

(아) GVM 플랫폼과 동일한 시스템 변수 및 API 제공

목적 시스템은, GVM에서 사용되는 아래와 같은 시스템 변수 및 API와 동일한 기능을 하는 변수 및 함수들을 제공해야 한다.

표 2. GVM의 시스템 변수

| 구분 | 변수명 | 기능 |
|---|---|---|
| Timer&Key | swData | Timer 이벤트와 Key 이벤트에 따른 구분 값 |
| 프레임 카운트 | swFrame | 0~1까지 반복되는 Frame Counter |
| | swFrame3 | 0~2까지 반복되는Frame Counter |
| | swFrame6 | 0~5까지 반복되는Frame Counter |
| LCD Size | swWidth | LCD 가로 크기 |
| | swHeight | LCD 세로 크기 |

표 3. GVM의 API

| 구분 | 주요 함수 |
|---|---|
| System | GetHwConfig |
| Graphic | ClearWhite, ClearBlack, Clear<br>PutPixel, SetGamma, SetColor, SetPalette<br>DrawLine, DrawHLine, DrawVLine, DrawRect, FillRect<br>DrawEllipse, FillEllipse<br>SetStrType, SetStrFont, SetStrColor, SetStrAlign<br>DrawStr, DrawText,<br>CopyImage, CopyImageDir, CopyImagePal, CopyImageDirPal<br>ScrollLCD, SaveLCD, RestoreLCD, Flush |
| HandSet Control | PlaySound, StopSound<br>StartVib, StopVib<br>SetBackLight<br>GetUserNV, PutUserNV<br>SetTimer, SetTimer1, SetTimer2<br>ResetTimer, ResetTimer1, ResetTimer2 |
| Mathematics | RandSeed, Rand, RandRatio<br>Abs, Sgn, Max, Min, ArrayToVar |
| Miscellaneous | HitCheck, GetDate, GetTime |

## 3. GVM2BREW의 자동변환도구

### 3.1 시스템 구성도

본 논문에서 제시할 GVM2KDDIBREW 자동변환도구의 전체 시스템 구성도는 그림 1과 같다.

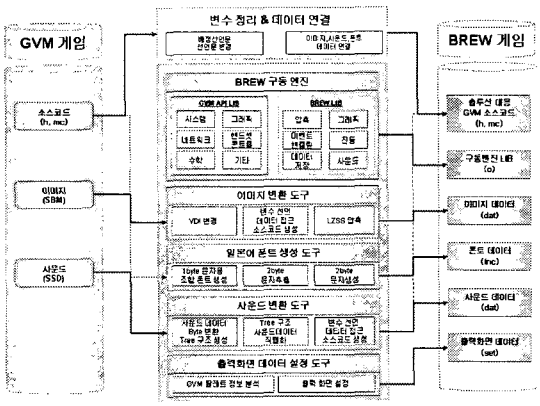



그림 3. GVM2BREW 자동변환도구 시스템 구성도

또한, 개별 요구사항에 대한 시스템 구성에 대한 방향성은 그림 4와 같다.

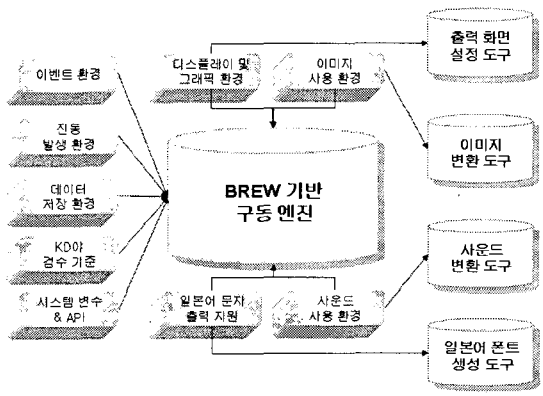



그림 4. 개별 요구 사항에 대한 시스템 구성 방향

### 3.2 요구사항분석 및 기반설계내용

3장의 분석된 요구사항을 기반으로 각 요구 사항을 만족시키기 위한 방법론 및 설계 방향을 모색하고, 통합적 관점에서 이를 재분석하여 설계한 기반설계내용은 다음과 같다.

1) GVM 플랫폼과 동일한 디스플레이 및 그래픽 환경 제공

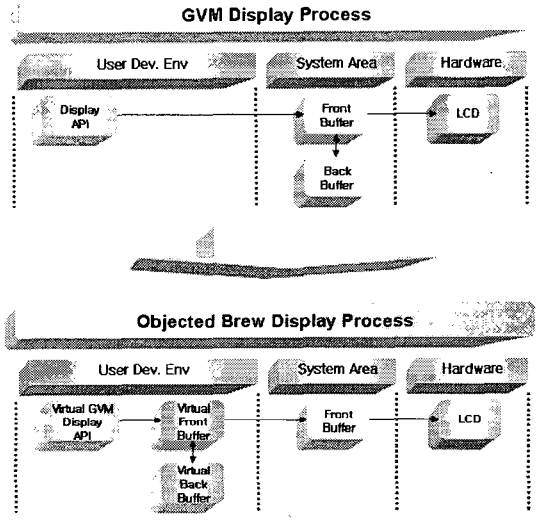



그림 5. GVM 디스플레이 처리 작업

GVM에서는 Display API에서 지시하는 그래픽 동작을 Front Buffer에 적용한 후 Front Buffer를 LCD에 출력한다. Brew에서 이와 동일한 기능을 구현하기 위하여, DIB(Device Independent BMP) 형태의 '가상 프론트 버퍼'와 '가상 백 버퍼'를 이용한다. GVM과 동일한 그래픽 Operation을 수행하는 Virtual GVM Display API를 통해 Virtual Front Buffer에 그래픽 동작을 적용한 후 Virtual Front Buffer를 변환하여, DDB 형태의 이미지를 생성하고, 이 이미지를 Front Buffer에 출력하는 방식을 사용하게 되는 것이다.

또한, 그림 5에서 보이는 바와 같이, GVM과 동일한 색상 및 Gamma Level이 적용된 색상을 사용할 수 있기 위한 BREW 환경에서의 가상 팔레트 테이블 생성 과정이 필요하다.

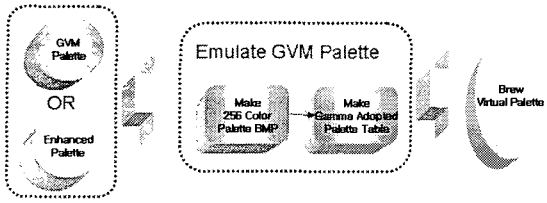



그림 6. GVM Palette 처리 작업

2) GVM 플랫폼과 동일한 이벤트 환경 제공

그림 7에서 보는바와 같이 GVM의 키 처리는 시스템이 전달한 이벤트를 '키 이벤트 핸들러 API'가 처리하는 방식이다. Brew에서 이를 구현하기 위하여, 시스템에서 전달한 이벤트를 'Brew 키 이벤트 핸들러'가 받아서 '가상 GVM 키 이벤트 핸들러 API'로 전달하도록 한다. GVM의 타이머 이벤트 처리는 등록된 타이머에 의해, 일정 시간 경과 후 시스템에서 발생된 타이머이벤트를 '타이머 이벤트 핸들러 API'가 처리하는 방식이다. Brew에서는, 동일한 기능 구현을 위해, 시스템으로부터 전달된 이벤트를 'Brew 타이머 이벤트 핸들러 API'에서 받은 후, '가상 GVM 타이머 이벤트 핸들러 API'에 전달하도록 한다.

3) GVM 플랫폼과 동일한 이미지 사용 환경 제공

GVM에서는 이미지 디스플레이 API를 통해 시스템에서 제공하는 VDI 디코더와 레스터 연산을 통한 그리기 작업을 통해서 이미지를 출력한다. Brew에서 이를 구현하기 위하여 VDI Decoder와 Raster Drawer를 구성하고 가상 GVM 이미지 디스플레이 API가 이들을 사용하도록 한다. 이러한 과정은 그림 8에서 보이는 바와 같다. 이를 위하여 GVM VDI 형식의 이미지를 BREW에서 사용할 수 있도록 압축하여 변환하는 기능을 필요로 하는데, 이러한 변환 과정을 도식화한 내용은 그림 9와 같다. 여기서, SBM 파일을 VDI Decoder를 거치게 하고, Original 256 컬러 팔레트 BMP와 Change 256 컬러 팔레트 BMP의 정보를 팔레트 정보 판독기를 거치게 하여 각각 이미지의 화소 정보와 색상 변환에 필요한 컬러 데이터를 모은 후, 이미지 컬러 변환기를 거쳐 이미지의 색상을 Change 256 컬러 팔레트 BMP에 맞추어 주고, VDI Encoder 후 데이터 압축기를 거쳐 주도록 구성되어 진다.

4) GVM 플랫폼과 동일한 사운드 사용 환경 제공

GVM에서는 Sound API가 시스템의 사운드 컨트롤러를 이용하여 사운드를 출력하게 한다.

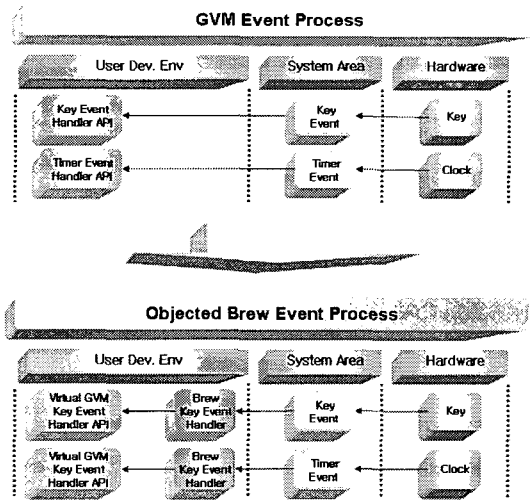



그림 7. GVM 이벤트 처리 작업

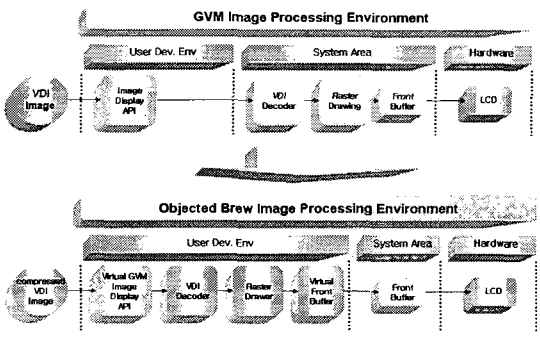



그림 8. GVM 이미지 처리 작업

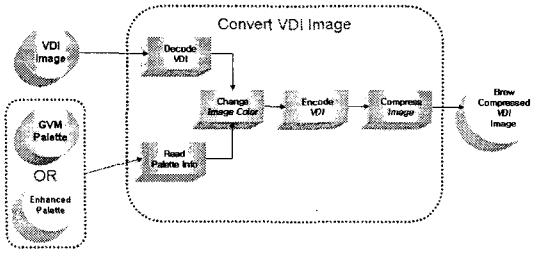



그림 9. VDI 이미지 변환

Brew에서 이를 구현하기 위하여 Brew 사운드 API를 사용하는 가상 GVM 사운드API를 구성하여 시스템의 사운드 컨트롤러를 이용할 수 있도록 한다.

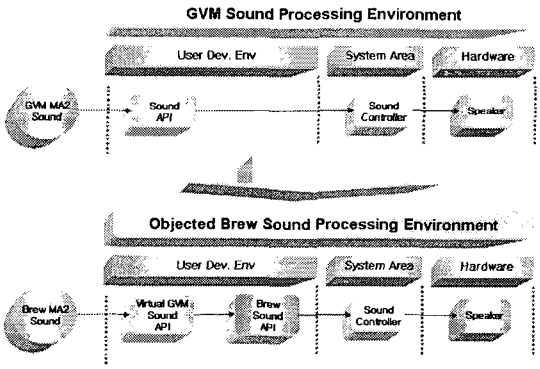



그림 10. GVM 사운드 처리 작업

이를 위하여 GVM의 SSD 파일에서 사운드 데이터를 추출하여 Brew Sound를 생성하는 과정이 그림 11과 같이 필요하다. SSD 파일을 디코딩하여 개별 사운드 데이터를 수집 및 병합하여 Brew 용 사운드 데이터로 변형하게 된다.

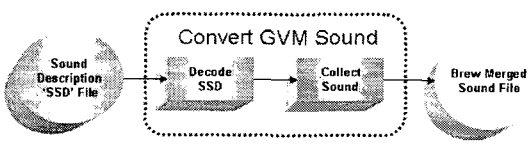



그림 11. GVM 사운드의 변환

5) GVM 플랫폼과 동일한 진동 발생 환경 제공

GVM에서는 바이브레이션 API가 시스템의 바이브레이션 제어기를 이용하여 바이브레이션을 출력하게 한다. Brew에서 이를 구현하기 위하여 Brew 바이브레이션 API를 사용하는 가상 GVM 바이브레이션 API를 구성하여 시스템의 바이브레이션 제어기를 이용할 수 있도록 한다.

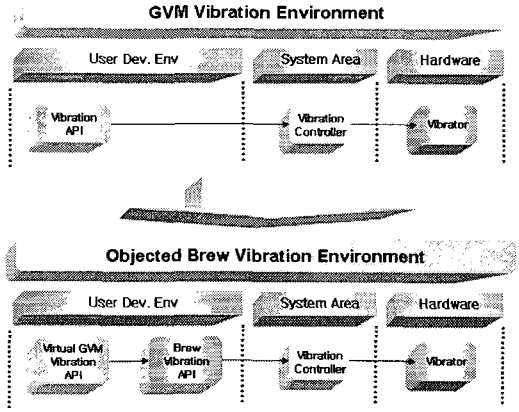



그림 12. GVM 바이브레이션 처리

6) GVM 플랫폼과 동일한 데이터 저장 환경 제공

GVM에서는 저장소용 API를 통해, 시스템의

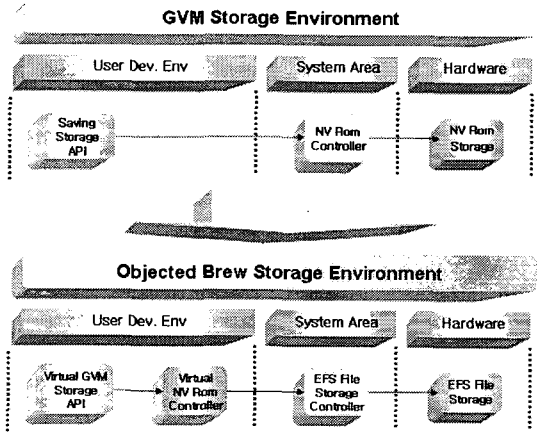



그림 13. GVM 스토리지 처리

NVRom에 사용자 데이터를 영구 저장하게 된다. Brew에서 이를 구현하기 위하여 시스템의 EFS 스토리지 제어기를 이용하여 가상 NV 롬 제어기를 구성하고 이를 사용하는 가상 GVM 저장 스토리지 API를 구성하도록 한다.

7) GVM 플랫폼의 한글과 동일한 규격의 일본어 문자 출력 환경 제공

GVM에서는 스트링 디스플레이 API를 통해서 폰트 탐색기에 출력 대상 문자를 입력하여 그에 알맞은 폰트 출력 정보를 찾은 다음, 폰트 Drawer를 통해 문자의 모양을 위한 폰트를 출력한다. Brew에서 이를 구현하기 위하여 가상 폰트 디스플레이 API를 구성한다. 가상 폰트 디스플레이 API를 통해 입력된 문자에 맞는 폰트 출력 이미지를 찾기 위한 폰트 탐색기를 구현 하고, 탐색된 폰트를 입력 문자의 바이트 구성에 맞추어 2 바이트(한글, 일본어) 폰트 출력하는 2 바이트 폰트 Drawer와 ASCII 문자를 출력하기 위한 1 바이트 ASCII 폰트 Drawer로 분류하여 가상 폰트 버퍼에 출력하도록 한다.

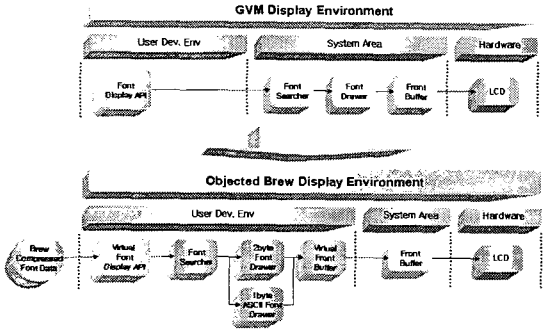



그림 14. 한글과 동일한 규격의 일본어 문자 출력 환경 구축

이를 위하여 폰트를 출력하는데 필요한 정보를 담은 압축된 폰트 출력 데이터가 필요하며, 이를 생성하기 위한 과정은 아래와 같다.

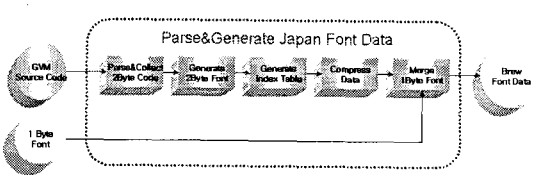



그림 15. 일본어 이미지 폰트 데이터 생성 과정

위 과정은 프로그램 소스 코드를 파싱하여 압축된 일본어 이미지 폰트 및 검색 데이터를 출력하기 위한 과정을 보여주고 있다. 2 바이트 일본어 문자들을 검색하고 검색된 일본어 문자와 일치하는 이미지들을 생성하여, 이 이미지들과 1 바이트 코드의 문자 이미지 폰트들을 결합한 후, 문자열 출력 시, 해당 일본어의 이미지 폰트를 검색하기 위한 인덱스 테이블을 생성하여, 압축하는 과정을 통해 결과물을 얻게 된다.

8) GVM 플랫폼과 동일한 시스템 변수 및 API 제공

게임에서 사용되는 GVM의 모든 API 함수들의 Wrapper 함수를 제공하여, 기존 GVM 소스코드의 변경 없이도, 동일한 기능을 수행할 수 있는 라이브러리를 제공한다.

### 3.3 GVM2BREW 시스템의 테스트

1) 테스트 방법론

GVM2BREW 시스템의 테스트는 자동 변환 소프트웨어의 특성에 입각하여, 그림 16에서와 같이 '점증적 접근 방식'을 통해 진행하였다. 먼저 가장 작은 단위의 단위모듈에 대한 '화이트 박스 테스트'를 통해 내부 논리구조의 오류를 발견하여 수정하고, '상향식 통합시험' 방법에 근거하여, 기능별 모듈에 대한 '블랙 박스 테스트'와 '기능 모듈간 연결성 테스트'를 수행한 이후, 시험 게임을 적용하여 소프트웨어 전체의 기능을 시험함으로써 테스트를 완료하였다.


그림 16. 테스트 진행 요약도

2) 테스트 게임의 변환

시험 대상 GVM 모바일 게임 '무사도' 그림 17~19를 본 연구 결과물인 자동변환도구를 통해 변환하고, 생성된 BREW 실행 파일이 GVM 게임과 동일한 기능 및 형태를 제공하며, KDDI 검수 기준에 적합한가를 아래와 같이 확인하였다.

변환된 결과물에 대한 안정성 및 성능에 대한 검증결과는 성공적이었다. 또한, GVM과 동일한


그림 17. 시험 게임 화면 1


그림 18. 시험 게임 화면 2

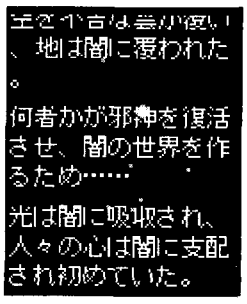



그림 19. 시험 게임 화면 3

표 5. 시험 게임 테스트 결과

| 게임명 | 분류 | | 시험 평가 항목 | 시험 횟수 | 성공 횟수 | 성공률 |
|---|---|---|---|---|---|---|
| 무사도 | 문법 | | 컴파일 정상작동 | 10 | 10 | 100% |
| | | | 누락된 GVM 함수 및 변수 | 없음 | | |
| | 그래픽 | Display 환경 | 화면 출력 정상 | 10 | 10 | 100% |
| | | | GVM Gamma 값 정상 작동 | 10 | 10 | 100% |
| | | 백터 | 점, 선, 원, 사각형등 GVM 그래픽 기능의 올바른 작동 | 10 | 10 | 100% |
| | | 이미지 | 이미지 데이터 변수 구성파일 생성 프로세스 정상 작동 | 10 | 10 | 100% |
| | | | 이미지 출력 역상 출력 팔레트 변경 출력 정상 작동 여부 | 10 | 10 | 100% |
| | | 문자 | Font 데이터 생성 | 10 | 10 | 100% |
| | | | 문자 출력 속성 변경 | 10 | 10 | 100% |
| | 사운드 | | 사운드 데이터와 변수 구성파일 생성 프로세스 정상 작동 | 10 | 10 | 100% |
| | | | 사운드 정상 출력 | 10 | 10 | 100% |
| | 진동 | | 진동 정상 출력 | 10 | 10 | 100% |
| | 저장 | | 읽기 쓰기 정상 작동 | 10 | 10 | 100% |
| | 사용자 이벤트 처리 | | 자판 입력 정상 작동 | 10 | 10 | 100% |
| | | | 콜수신, 비정상 종료 시 정상 작동 | 10 | 10 | 100% |

기능을 제공하기 위한 변환 시간은 총 3시간이었으며, KDDI 규격을 위한 일본어 및 UI 변경 작업에 4일(32시간)의 수행시간이 소요되었다. GVM에서 KDDI-BREW 플랫폼으로 일반적인 변환 작업에 의해서 진행했을 때 1개월~3개월의 시간이 걸리는 것에 비교하면, 10배 이상의 시간을 절감할 수 있었다.

## 4. 결 론

본 논문에서는 GVM 게임을 KDDI-BREW로 자동으로 변환하는 도구를 개발하였다. 이를 통해, 모바일 게임 멀티 플랫폼 변형 솔루션 개발을

위한 연구의 기반을 형성하였으며, 기존에 GVM 플랫폼으로 개발되었던 시험 게임을 일반적인 변환 과정을 거쳤을 때의 시간보다 10배 이상 단축하여 개발하는데 성공함으로써, 결과물에 대한 안정성과 신뢰성을 확보하였다. 현재의 완성물은 자동 변환 도구가 개별 도구로 분리되어 통합화되어 있지 않으며, 변환 작업 시 사용자의 개입에 의해서 처리되는 프로세스가 남아있는 등의 단점 을 가지고 있으나, 목표로 하였던 변환 시간의 단축 및 변환 결과물의 안정성에 있어서는 기대 이상의 성과를 가지게 되었다.

본 자동 변환 도구에 대한 연구 개발을 해, 모바일 게임 멀티 플랫폼 변형 솔루션을 위한 하나의 성공적인 모델이 정립되었다. 현재의 개발 결과물에 상용화를 위한 기능 및 자동화를 위한 기능이 보완되어 출시될 경우, 수출이 가속화되어가는 모바일 게임 산업의 현황에서, 개발사들이 보다 빠른 시간에 자사가 개발한 게임을 일본으로 수출하기 위한 기술적인 기반을 마련하게 될 것이며, 새로운 시장인 일본을 공략하기 위한 글로벌 콘텐츠 기획 및 개발에 박차를 가하게 될 것이다.

또한, 본 연구 개발 성과에 따라 다음과 같은 추가 연구 개발의 진행이 가속화 될 것으로 예상된다. 첫째, '통합 솔루션 개발'이다. 본 과제의 결과물들을 하나로 개발 도구로 통합되어, 통합 솔루션 내에서 모든 기능을 자동화된 형태로 진행할 수 있도록 함과 동시에, 보안 인증, 배포 등이 용이한 형태로 개발되어, 확장성을 확보 하고 향상된 기능과 사용자 편의성을 제공하여, 상용화가 가능해질 것이다. 둘째, '타 해외 BREW 솔루션 개발'이다. 미국 Varizon, 남미 Vibo, 중국 China Unicom 등의 이동통신사에서 채택한 Brew 플랫폼 향으로의 변환 솔루션을 추가로 개발하여, 여러 가지 BREW 플랫폼에 적용될 수 있는 GVM2BREW 통합 솔루션 구축이 가능해 질 것이다. 이를 위해서는 위해 플러그인 형태의 기능 추가가 가능해야 할 것이다. 셋째, 'GVM2MIDP 솔루션 개발'이다. GVM으로 개발된 모바일 게임을 전 세계적으로 가장 많은 핸드폰에서 탑재하고 있는 MIDP 형태로 변형하는 솔루션을 개발하게 될 것이다. 마지막으로 'MIDP2MIDP 솔루션 개발'이다. MIDP(Java)로 개발된 모바일 게임을 해외의 다양한 MIDP 플랫폼으로 자동 변형하는 솔루션을 개발하게 될 것이다.

## 참 고 문 헌


[1] 김필진, 나승원, 고석훈, 오세만, "Mobile Game Programming using GVM and MobileC", IASTED International Conference Communication and Computer Networks, pp. 451~456, 2002.

[2] 모바일 브루 사이트 운영진, BREW Mobile Programming, 영진.COM, 2002.

[3] 신지소프트, 위피 게임 서비스 어플리케이션 GNEX™ Technical White Paper, 2003.

[4] 신지소프트, GNEX SDK, http://www.gnexclub.com/download/ download.jsp, 2004.

[5] 앤슬래시닷컴, GVM Programming: General Virtual Machine Programming, 삼양출판사, 2001.

[6] 엠버튼, 모바일 게임 콘텐츠 멀티플랫폼 적용 변환 솔루션 개발 보고서, 한국게임산업개발원, 2005.

[7] 이 양선 외 2인, "JVM 플랫폼에서 .NET 프로그램을 실행하기 위한 MSIL-to-Bytecode 번역기의 설계 및 구현," 한국멀티미디어학회 논문지, Vol.7, No.7, pp.976-984, Jun 2004.

[8] 이 양선, 외 2인, "Intermediate Language Translator for Execution of Java Programs in .NET Platform," 한국멀티미디어학회 논문지, Vol.7, No.6, pp.824-831, Jun 2004.

[9] 이정환, 소주호, GVM & MAP 모바일 프로그래밍, 대림, 2002.

[10] 정찬성, "WIPI 게임 솔루션 GNEX를 이용한 모바일 게임 제작", 정보과학회 튜토리얼, http://www.ioi2002.or.kr/conference/2004s/source/intro/T3.ppt, 2004. 04.

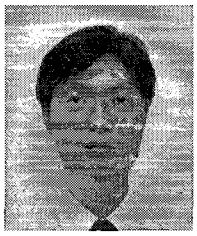
[11] 천귀호, BREW 모바일 프로그래밍, 한빛미디어, 2002. 05.

[12] Qualcomm "BREW™ and J2ME - A Complete Wireless Solution for Operators Committed to Java", http://brew.qualcomm.com/brew/en/about/white_papers.html.

[13] Qualcomm "The Road to Profit is Paved with Data Revenue", http://brew.qualcomm.com/brew/en/about/white_papers.html.

[14] Qualcomm "BREW™ Application Extensions Overview" 80-D4190-2 Rev.A, http://brew.qualcomm.com/brew/en/developer/resources/ad/extensions.html.

[15] YangSun Lee, SeungWon Na, "Java Bytecode-to-.NET MSIL Translator for Construction of Platform Independent Information Systems," LNAI 3215, Vol. 3, ISSN 0302-9743, Springer, pp.726-732, Sep 2004.



조 준 호

- 1999년 서강대학교 컴퓨터공학과(공학사)
- 1999년~2001년 ㈜디비엠코리아 선임연구원
- 2001년~현재 (주)테크론시스템 대표이사
- 관심분야 : 모바일 온라인 게임 개발, 모바일 콘텐츠 자동 변환 솔루션, 네트워크 솔루션 등

홍 철 운

- 2002년 삼육대학교 영문학과(문학사)
- 1999년~2000년 카이소프트 대표이사
- 2000년~2001년 ㈜레이넷 개발총괄팀장
- 2001년~현재 ㈜엠버튼 대표이사
- 관심분야 : 모바일/온라인/아케이드 게임개발, 모바일 콘텐츠 자동변환 솔루션, 게임품질관리 등

이 양 선

- 1985년 동국대학교 전자계산학과(공학사)
- 1987년 동국대학교 대학원 컴퓨터공학과(공학석사)
- 1993년 동국대학교 대학원 컴퓨터공학과(공학박사)
- 1994년~현재 서경대학교 컴퓨터공학과 교수
- 1996년~2000년 서경대학교 전자계산소 소장
- 2000년~2004년 멀티미디어학회 이사
- 2005년~현재 멀티미디어학회 총무이사
- 2004년~현재 인터넷정보학회 이사
- 2004년~2005년 한국정보처리학회 게임연구회 부위원장
- 2005년~현재 한국정보처리학회 게임연구회 위원장
- 관심분야 : 프로그래밍언어, 임베디드 시스템, 모바일 컴퓨팅, 게임 기술, CT 기술 등