GenRoom AI

Software Requirements Specification(SRS)

과목명: 졸업 프로젝트
팀원 : 안재민, 김인교, 노을영
담당교수: 유준범
제출일: 2026.05.12

Table of Content

1. Introduction

1.1. Purpose

- 본 문서는 3D 공간 자동 생성 시스템인 'GenRoom AI'의 SRS 이다. 이 문서는 시스템 유지보수팀 및 개발팀이 시스템의 주요 기능 및 개발 범위를 이해하고, 이를 기반으로 확장성 높은 시스템을 만드는 데 필요한 정보를 제공한다.

1.2. Scope

- 본 시스템은 'GenRoom AI'로 사용자로부터 자연어 입력을 받아 공간 내 가구의 좌표값과 3D Mesh를 생성하고 Unity를 통해 방을 생성하는 것까지 구현한다. 시스템은 UI Layer, Positioning Layer, 3D Modeling Layer, Generation Layer로 구성된다. UI Layer는 사용자의 입력을 받아오고, Positioning Layer는 공간 내 가구 정보를 생성하고, 3D Modeling Layer는 가구 3D Mesh를 생성하고, Generation Layer는 결과물들을 통합해 최종 공간을 출력한다.
- 이 SRS는 GenRoom AI의 소프트웨어 요구사항에 대한 전반적인 설명을 포함하며, UI Layer는 User Interface로 작성하고 다른 Layer에 대해서 Functional Requirement로써 자세하게 작성한다.

1.3. Definitions, acronyms and abbreviations

- GenRoom AI: 본 프로젝트의 명칭이자 시스템 이름
- UI Layer: User의 자연어 입력을 받아오는 Layer
- Positioning Layer: User의 입력을 바탕으로 공간의 구성 정보를 생성해내는 Layer, 해당 Layer에서 생성된 값에 대한 Validation 또한 수행한다.
- 3D Modeling Layer: Positioning Layer에서 나온 결과값들을 바탕으로 3D Mesh를 생성해내는 Layer. Positioning Layer에서 의도한 대로 Mesh 생성되었는 지 확인하는 작업 또한 수행한다.
- Generation Layer: Layer 결과물을 통합해 사용자에게 출력해주는 Layer
- Bounding Box(BB): 가구 배치를 위한, 가구가 들어갈 수 있는 직육면체 공간

1.4. References

- IEEE Std 830-1998: IEEE Recommended Practice for Software Requirement

1.5. Overview

- 제 1장: 시스템 개요 및 문서 목적 설명
- 제 2장: 시스템 주요 기능 및 프로젝트 전제 환경 설명
- 제 3장: 각 컴포넌트별 기능 요구사항 세부 설명. Layer를 기준으로 세분화하여 제시한다.

2. Overall description

2.1. Product perspective

- GenRoom AI는 사용자의 자연어 입력을 통해 가구 Mesh와 함께 방을 자동 생성해주는 시스템으로서, 기존 절차적 방 생성 프로그램의 정형적 생성 문제를 해결하고, AI가 제시하는 방의 좌표 수치값을 알맞게 조정하는 파이프라인을 설계, 구현하는 것을 목표로 한다.
- 이 시스템은 4개의 주요 Layer로 구성되며, 각 Layer는 정형화된 데이터를 Output으로 산출하여 다음 Layer의 Input으로 사용하게 된다.
- 기존에 구현되어있는 LLM, 방 배치 AI(DiffuScene 등), 3D Mesh 생성 AI, Unity Engine을 각각 사용하는 4개의 Layer로 구성되어 있으며, 각 Layer의 결과값을 우리가 요구하는 이상치에 가깝게 후처리하는 것을 목표로 한다.

2.2. Product functions

- 자연어 입력**:** 사용에게 자연어 입력을 받아 어떤 방을 만들것인지(테마, 크기, 가구 수등)을 판단하여 방 배치 AI에게 제시한다.
- 좌표 이상치 조절**:** 방 배치 AI가 제시한 가구들의 좌표값이 이상이 있는지 판단 후 그를 해결하는 알고리즘을 실행한다.
- 생성된 **3D Mesh** 이상 판단**:** 3D Mesh 생성 AI를 통해 생성한 Mesh가 방 배치 AI가 의도한 대로 생성되어 있는지 판단 후 문제를 해결하는 알고리즘을 실행한다.
- 방 생성**:** 각 Layer에서 검증된 결과값들을 바탕으로 사용자에게 가구들이 배치된 완성된 방을 제공한다.

2.3. User characteristics

- 월드를 생성하는데 많은 시간을 투자하고 싶지 않은 '인디 게임 개발자'들을 대상으로한 시스템이다.

2.4. Constraints

- 오픈소스 AI를 활용하여 저작권법에 위촉되지 않는 시스템을 개발한다.
- 3D Mesh 생성 AI를 사용하여 모델을 생성하는 시간은 1개당 최대 1분이내로 진행되어야한다.
- 최종 3D 모델 파일은 GLB, FBX형식을,좌표 파일은 Json 형식을 준수해야한다.
- Gemini API 등 외부 서비스의 가용성 및 네트워크 연결 상태에 따라 시스템 기능이 제한될 수 있다.
- 각 Layer간 상호 간섭을 최소화하기 위해 Json 포멧을 통해서만 데이터를 주고받아야한다.
- 가구 배치 알고리즘은 Python 환경에서 구현하며, 최종 시각화 및 런타임 환경은 Unity엔진 사용으로 제한한다.
- 시스템에 사용되는 모든 외부 API 키 및 인증 토큰은 소스 코드 내에 직접 노출되지 않도록 환경 변수 등을 통해 보안 관리되어야 한다.

2.5. Assumptions and dependencies

- 본 시스템은 기존에 구현되어 있는 각종 API들에 의존하여 작동하는 시스템이며, 우리가 제시한 API 외 비슷한 기능의 API로 대체하였을 때에도 정상 작동하는 것을 목표로 한다.

3. Specific requirements

3.1. External interface requirements

3.1.1. User Interface

3.1.1.1. 사용자 입력 메인 화면

3.1.1.2. 시스템 메세지 출력 창

3.1.1.3. 시스템 진행 상황 출력 화면

3.1.2. Software interfaces

- UI Layer to Positioning Layer Data 명세서(Json 형식)

```
{
"room_type": "BedRoom",
"room_size": { "x": 6, "y": 5, "z": 3 },
"atmosphere": "Horror",
“furniture number”: 7,
}
```

- Positioning Layer to 3D Mesh Layer Data 명세서

```
{
"room_type": "BedRoom",
"room_size": { "x": 6, "y": 5, "z": 3 },
"atmosphere": "Horror",
“furniture_number”: 7,
"objects": [
   {
     "name": "stone_fireplace",
     "position": {
       "x": -4.15,
       "y": -1.15,
       "z": -2.10
     },
     "rotation": {
       "x": 0.0,
       "y": 90.0,
       "z": 0.0
     },
     "scale": {
       "x": 1.75,
       "y": 2.70,
       "z": 0.95
     }
   }]
}
```

- 3D Mesh Layer to Generation Layer Data 명세서

```
{
"room_type": "BedRoom",
"room_size": { "x": 6, "y": 5, "z": 3 },
"atmosphere": "Horror",
“furniture number”: 7,
"objects": [
   {
     "name": "stone_fireplace",
     “file_name”: “stone_fireplace_mesh.glb”,
```

```
    "position": {
      "x": -4.15,
      "y": -1.15,
      "z": -2.10
    },
    "rotation": {
      "x": 0.0,
      "y": 90.0,
      "z": 0.0
    },
    "scale": {
      "x": 1.75,
      "y": 2.70,
      "z": 0.95
    }
   }]
}
```

3.1.3. Communication interfaces

- 각 Layer의 Communication 간에는 Data를 정해진 Json 형식에 따라 UTF-8로 인코딩하여 전달한다.
- Gemini API와 같은 외부 생성형 AI 서비스와 통신할 때는 HTTPS/REST API를 사용한다.

3.2. Functional Requirement

3.2.1. UI Layer

3.2.1.1. 사용자 입력 처리

3.2.1.1.1. 사용자로부터 자연어 형태의 Input 데이터를 받아야 한다.

3.2.1.1.2. 공백, 특수문자, 의미 없는 텍스트 입력 시 시스템이 중단되지 않고 적절한 오류 메시지를 반환하거나 재입력을 유도해야 한다.

3.2.1.2. 자연어 사용자 입력기반 프롬포트 작성

3.2.1.2.1. 사용자로부터 입력받은 일반 텍스트를 분석하여 표준화된 프롬프트로 변환해야 한다.

3.2.1.3. **LLM** 출력값 **Json** 형식으로 전처리

3.2.1.3.1. 유효한 출력값에서 가구 종류 및 수 등의 변수를 정수(Integer) 형태로 추출하여 Json 형식으로 정리해 후속 레이어로 전달해야 한다.

3.2.1.4. **API** 통신 및 응답 처리

3.2.1.4.1. 생성된 프롬프트를 통해 API를 호출하고, 수신된 JSON 형태의 응답 데이터가 유효한지 체크해야 한다.

3.2.1.5. **.Json** 파일 저장

3.2.1.5.1. 상기 생성된 .Json 파일을 로컬에 저장한다.

3.2.2. Positioning Layer

3.2.2.1. **Json** 형식 **Data** 기반 배치 **Model** 사용

3.2.2.1.1. 입력된 가구 카테고리 정보를 바탕으로 각 가구의 위치, 회전값 생성해야 한다.

3.2.2.2. 좌표 정보 오류값 처리

3.2.2.2.1. 생성된 좌표가 바닥이나 벽을 관통하지 않도록 자동으로 보정 작업을 수행해야 한다.

3.2.2.3. 좌표 정보 **Validation**

3.2.2.3.1. **Collision** 판단

3.2.2.3.1.1. 두 가구 간의 Bounding Box가 겹칠 경우, 단순한 면 접촉(Edge Contact)은 허용하되 실제 충돌 시에는 반복(Iteration)을 통해 위치를 재산출해야 한다 .

3.2.2.3.2. 가구 수 판단

3.2.2.3.2.1. 사용자가 입력한 가구의 수에 맞게 가구가 배치되어 있는가 판정해야 한다.

3.2.2.3.3. 반복 제한 및 예외 처리

3.2.2.3.3.1. 가구 배치가 불가능한 한정된 공간에 수용가능한 개수 이상의 가구가 배치되는 경우 무한 루프에 빠지지 않도록 최대 반복 횟수를 제한하고, 초과 시 사용자에게 알림을 주거나 가능한 부분만 배치해야 한다

3.2.2.3.4. 방과 가구 크기의 정합성

3.2.2.3.4.1. 입력된 방의 크기 대비 과도하게 큰 가구 배치가 요청될 경우 이를 인식하고 예외 처리해야 한다

3.2.2.4. **.Json** 파일 저장

3.2.2.4.1. 3.2.1.5에 의해 생성된 .Json 파일에 새로운 Data를 추가하여 덮어쓰기 형식으로 써서 저장한다.

3.2.3. **3D Modeling Layer**

3.2.3.1. **3D Mesh** 생성 프롬프트 작성

3.2.3.1.1. UI 및 Positioning Layer에서 넘어온 정보를 바탕으로 적절한 가구 3D 메시 파일을 생성하기 위한 프롬프트를 각각 작성한다.

3.2.3.2. **3D Mesh AI**를 통한 **Mesh** 생성

3.2.3.2.1. 제작한 프롬프트를 기반으로 각 가구에 최적화된 3D 메시 파일(.glb, .obj 등)을 생성해야 한다.

3.2.3.3. **3D Mesh Validation** 판단

3.2.3.3.1. 스케일 검증

3.2.3.3.1.1. 생성된 3D 모델의 실제 크기가 Positioning Layer에서 정의한 Bounding Box 스케일과 일치하도록 스케일 수정 기능을 수행해야 한다.

3.2.3.3.2. 방향 검증

3.2.3.3.2.1. 생성된 3D 모델의 방향이 Positioning Layer에서 정의한 방향과 일치하도록 모델의 방향 값 수정 기능을 수행해야 한다.

3.2.3.3.3. 결함 에러 해결

3.2.3.3.3.1. AI 모델 생성 실패나 네트워크 타임아웃 발생 시, 시스템 흐름 유지를 위해 미리 정의된 기본(Default) 가구 모델로 대체 배치해야 한다

3.2.3.4. **.Json** 파일 저장

3.2.3.4.1. 3.2.1.5에 의해 생성된 .Json 파일에 새로운 Data를 추가하여 덮어쓰기 형식으로 써서 저장한다.

3.2.4. Generation Layer

3.2.4.1. 출력 엔진 실행

3.2.4.1.1. 최종 배치가 완료된 3D 방 환경을 사용자에게 실시간으로 렌더링하여 출력해야 한다.

3.2.4.1.2. 수신된 JSON 기반의 위치/회전 정보와 3D 모델 파일을 Unity 환경 내에서 1:1로 매칭하여 오차 없이 렌더링해야 한다

3.2.4.2. 좌표 정보 기반 바운딩 박스 배치

3.2.4.2.1. 수신된 좌표정보들을 바탕으로 각각의 좌표와 회전값에 맞춰 Bounding Box를 배치해야 한다.

3.2.4.3. 바운딩 박스 내 **3D Mesh** 배치

3.2.4.3.1. 수신된 3D 모델들을 각각의 Bounding Box에 맞춰 Unity 런타임 환경 내에 배치해야 한다.

3.2.4.4. 배치 상태 **Validation** 판단

3.2.4.4.1. 최종 배치 단계에서 시스템적으로 오류가 발생할 경우 재시도해야 한다.

3.2.4.4.2. 모든 레이어를 거친 후 최종 출력된 3D 방 환경은 가구 간 물리적 간섭이 없어야 하며 사용자에게 시각적 결함 없이 출력되어야 한다.

3.3. Performance requirements

- 하나의 3D Modeling을 생성하는데, 1분을 초과하지 않도록 한다.
- 하나의 방 배치 좌표를 생성하는데, 5분을 초과하지 않도록 한다.
- 생성된 Data를 종합하여 전체 방을 사용자에게 출력하기까지 1분을 초과하지 않도록 한다.

3.4. Design constraints

- 해당 시스템은 실행파일(프로그램)의 형태로 사용자에게 제공되며 사용자는 파일을 실행하는 것으로 사용할 수 있도록 한다.
- 각 Layer간의 Data 전달은 Json 파일을 주고 받는 것으로 구현한다.
- 게임 월드 생성을 1차적인 목표로 두고 있으므로 Generation Layer에서 실행시키는 출력 엔진은 Unity를 사용한다.
- UI Layer의 UI는 Unity 기반으로 작성한다.
- 생성된 Position들은 Json 형식으로 제공되며, 각 Mesh들은 FBX, GLB파일로 제공된다.
- 시스템은 Layer의 책임을 명확히 구분하여 한 Layer에서 발생하는 오류가 다른 곳에 영향을 미치지 않도록 구현해야한다.
- Positioning Layer 및 3D Modeling Layer에서는 AI와의 결합을 위해 Python을 통해 작성한다.

3.5. Software system attributes

3.5.1. 보안성**(Security)**

3.5.1.1. **API** 보안 **:** 외부 API(Gemini API 등) 통신 시 API Key 및 인증 토큰은 안전하게 관리되어야 하며 노출되지 않도록 처리해야 한다.

3.5.1.2. 이상 입력 처리**:** 시스템이 처리할 수 없는 입력이 들어오거나 개인정보에 접근하는 입력이 들어오면 시스템 내에서 에러로 처리해야 한다.

3.5.2. 가용성**(Availability)**

3.5.2.1. 결함 허용**(Fault Tolerance):** 3D 모델 생성 실패나 특정 레이어에서 오류가 발생하더라도 전체 시스템이 중단(Crash)되지 않고, 기본(Default) 모델 대체 등의 예외 처리를 통해 서비스를 지속해야 한 기본(Default) 모델 대체 등의 예외 처리를 통해 서비스를 지속해야 한다.

3.5.3. 사용성**(Usability)**

3.5.3.1. 직관적 인터페이스**:** 단순 자연어 입력창만을 UI로 제작하여 전문적인 지식이 없는 일반 사용자도 방 배치 결과물을 얻을 수 있도록 간결하게 설계해야 한다.

3.5.3.2. 피드백 제공**:** AI 모델의 생성 과정(이미지 생성, 배치 최적화 등) 동안 사용자가 대기 상태를 인지할 수 있도록 로딩 바나 안내 메시지를 제공해야 한다.

3.5.4. 신뢰성**(Reliability)**

3.5.4.1. 출력 데이터 정합성**:** UI에서 입력한 카테고리가 최종 3D 씬과 일치해야 하며, Bounding Box와 모델의 스케일이 일관성을 유지해야 한다.

3.5.4.2. **API** 응답 정합성**:** 비슷한 기능을 하는 API에 대해서 바꿔 사용하였을때 출력되는 값은 일정하게 나오도록 한다.

3.5.4.3. 예외 처리**:** 각 주요 Layer는 발생할 수 있는 error들에 대해 독립적인 예외 처리 로직을 갖추어야 한다.