公立はこだて未来大学 2025 年度 システム情報科学実習
グループ報告書

Future University Hakodate 2025 Systems Information Science Practice
Group Report

プロジェクト番号/Project No.

10

グループ名/Group Name

はこぺパッと / Hakopepat

プロジェクト名

シン・函館補完計画「棒二森屋」跡地をフィールドとする地域共創 AR サービスのデザイン

Project Name

Design of a Community Co-Creation AR Service Based in the Former Site of Boni-Moriya

プロジェクトメンバー/Group Member

生田立樹　Ikuta Riki

亀貝奈桜　Kamegai Nao

北澤陽　Kitazawa Akira

村田藍途　Murata Aito

綿谷礼花　Wataya Ayaka

指導教員

安井重哉　松原克弥　高見逸平

Advisor

Yasui Shigeya　Matsubara Katsuya　Takami Ippei

提出日

2025 年 1 月 20 日

Date of Submission

January 20, 2025

概要

本報告書は，函館駅前地区の象徴であった棒二森屋跡地周辺の賑わい創出を目指し，AR（拡張現実）技術とペーパークラフトを融合させた「はこペパッと」の企画，設計，および初期検証結果をまとめたものである．JR 函館駅前に位置する棒二森屋百貨店は 2022 年に閉店し，2025 年現在まで周辺の賑わいは低下したままであり，地域課題となっている．本プロジェクトでは，「『棒二森屋跡地』をフィールドとする地域共創 AR サービスのデザイン」をテーマに掲げ，市民と観光客双方が魅力的に感じることを目的とした，AR 技術を利用したサービスを制作することを目指した．本グループでは，5 月に実施したフィールドワークでの「駅前なのに若者が行きたい店が少ない」という気づきを出発点とし，ワークショップを通じて理想の街をデザインするサービス「はこペパッと」を開発した．本サービス「はこペパッと」は，ペーパークラフト（紙の箱）に絵を描き，それを AR/VR 技術でデジタル空間に取り込むことで，創作体験とデジタル体験を融合させたものである．ユーザーが手書きした建物のデザインをスマートフォンで読み込むことで，仮想空間上で街に配置し，その街を散策できるサービスを提供する．複数回のプレワークショップとユーザーフィードバックを通じて，デジタル完結ではなく「箱を用いたリアルな体験」を重視する方向へ改善した．最終的なサービスは，「紙× AR の新感覚クラフト体験」「手軽で自由度の高いまちづくりデザイン」「未来を共創するまちづくり支援」の 3 つの価値を提供し，誰もが街の未来に参加できる仕組みを通じて，地域の賑わいを創出することを目指す．本報告書では，サービスの概要，開発プロセス，および今後の展望について述べる．

キーワード　AR，地域共創，まちづくり，ペーパークラフト，ワークショップ

Abstract

This report summarizes the planning, design, and initial evaluation results of "Hakopé Patto," a project that integrates AR (Augmented Reality) technology with paper craft to revitalize the area surrounding the former Boni Moriya department store site, which once served as a symbol of the Hakodate Station front district. The Boni Moriya department store, located in front of JR Hakodate Station, closed in 2022, and as of 2025, the surrounding area has continued to suffer from a decline in activity, making it a significant local issue. This project was conducted under the theme of "Designing a community co-creation AR service using the former Boni Moriya site as a field," with the goal of creating an AR-based service that would be attractive to both local residents and tourists. The project began with insights gained from fieldwork conducted in May, which revealed that "despite being in front of the station, there are few places that young people want to visit." Based on this realization, the team developed "Hakopé Patto," a service that allows users to design their ideal city through workshops. "Hakopé Patto" combines creative and digital experiences by allowing users to draw designs on paper crafts (paper boxes) and import them into a digital environment using AR/VR technology. By scanning hand-drawn building designs with a smartphone, users can place them within a virtual city and freely explore the space. Through multiple pre-workshops and user feedback sessions, the service was refined to emphasize tangible, real-world interaction using physical paper boxes rather than a purely digital experience. The final service delivers three core values: (1) a novel paper × AR craft experience, (2) accessible and highly flexible city-design creation, and (3) support for future-oriented community co-creation. By enabling anyone to participate in shaping the future of the city, the project aims to foster local vibrancy and engagement. This report describes the service overview, development process, and future prospects.

Keyword AR, community co-creation, urban design, paper craft, workshop

目次

第 1 章　はじめに

1.1　背景

棒二森屋百貨店は JR 函館駅前に位置していたが，2022 年の閉店以降，2025 年現在に至るまで周辺地域の賑わいは低下したまま推移しており，地域にとって解決すべき課題となっている．本プロジェクトメンバーが 5 月に函館駅周辺で実施したフィールドワークにおいて，「駅前という立地であるにもかかわらず，大学生などの若年層が自ら行きたいと思う店舗が少ない」という現状が確認された．この主観的な気づきから，実際にこの場所を訪れる人々がどのように感じているのか，また潜在的に何を求めているのかを明らかにする必要性が生じた．

1.2　目的

本プロジェクトは，「『棒二森屋跡地』をフィールドとする地域共創 AR サービスのデザイン」をテーマとしている．前述の背景を踏まえ，ワークショップを通じて参加者の潜在的なニーズや「本当に欲しいもの」を集約し，可視化することを目的とする．単なるアンケートではなく，体験型のツールを用いることで，これからの函館市のまちづくりに具体的に反映可能なアイデアを創出することを目指す．

第 2 章　プロダクトの事前調査・考案方法

2.1　事前調査

函館駅前周辺で利用してもらうサービスを提案・実装するため，ユーザーや函館駅前周辺に関する地理的・文化的な情報などについての理解を深めることが必要とされた．そこで，フィールドワークとインタビューを実施し，シーンについての解像度を高めた．

2.1.1　フィールドワーク

私たちは二回にわたりフィールドワークを行った．フィールドワークとは，プロダクトを配置する場所に実際に行き，関係者からの話を聞くなどの体験を通して，どのような現状や課題があるかを見つけ出して制作の参考にする作業のことである．

第一回フィールドワークでは函館駅前にどのようなものがあり，どのような雰囲気であるかを調べるため，メンバー間の親睦を深めるために行った．それぞれの感覚で駅前を探索し，写真を撮り記録した．第二回フィールドワークでは実装するときに活用したいもの，事前に考えておきたいことをそれぞれの視点で想定して行った．具体的には，大門エリアの古くからある店舗が昔の棒二森屋に対して抱いていた感情や，再活性化への関心について回答を求めること．そして，ロケーションベースド AR を配置するのに活用できそうな場所の探索を行った．

この結果，高齢者からは，棒二森屋周辺は昔は活気のあった地域だったが，街の明かりが減って賑わいが減少し，今後の発展には興味がないという回答があった．一方で，学生からは「娯楽施設やチェーン店が少ない」といった，棒二森屋周辺が学生の楽しめる場所ではないという回答があり，賑わいを取り戻してほしいという意見もあった．これらの発見・考察がアイデア発案につながった．

2.1.2　インタビュー

旧棒二森屋跡地の街づくりについてよく知っている市役所職員 2 名を大学に招き，インタビューを実施した．また，本学を数年前に退職した岡本先生にもインタビューを行った．市役所職員からは，棒二森屋跡地の発展に関する現在進行中の建設案と，それに関する地権者や利害関係者，行政としての立場について，また現状の棒二森屋跡地周辺に存在する駐車場とその利用状況についての知識を得た．岡本先生からは，プロダクトを利用してもらう地域や，そこに住む人々と共にプロダクトを制作する「共創」の考え方について助言を受け，地域に根ざした価値創造こそがプロジェクト成功の糧になるといった，今後の活動方針やプロトコルを制作するうえで有意義な意見をいただいた．

2.2　研修

本プロジェクトを円滑に進め，より質の高いプロダクトを考案・実現するために，以下の研修や学習会に参加した．まず，高見先生によるインタビュー法のレクチャーを受講した．これにより，

効果的な質問の組み立て方や，対象者から深掘りした情報を引き出すための技術を学んだ．質問を組み立てる流れはテーマや目的によって異なるが，5 から 7 のカテゴリーに分け，それぞれ 3 から 5 つの質問で構成する．広いフォーカスのカテゴリーから始めて具体的なカテゴリーに進むと，対象者が「何について回答すればいいのか」が分かりやすい流れになり，効果的なインタビューを行うことができる．これは，函館市役所職員や岡本先生へのインタビュー，そしてフィールドワーク中の簡易インタビューにおいて，より質の高い情報を得る上で不可欠な知識となった．

本プロジェクトが AR 技術の活用を核としているため，AR に関するリサーチを行った．既存の AR を用いた地域活性化事例や，様々な AR 機能の実装可能性について調査した．これにより，提案するサービスが技術的に実現可能であるかを検討し，サービス考案のための基盤を築いた．

個々人が発想した多岐にわたるアイデアを整理するための KJ 法の勉強会を行った．KJ 法とは文化人類学の分野で川喜田二郎が考案した研究法である．具体的には，アイデアや情報をカードに書き出し，グループ化や並べ替え，図解化などを行うことで，問題解決や新たなアイデアの創出を促す手法である．この手法を学ぶことで，アイデア出しの際に得られた膨大な情報をグルーピングし，関連性を分析し，本質的な課題やコンセプトを抽出する能力を養った．

プロジェクトの進行を効率化し，変化に柔軟に対応できる体制を構築するため，アジャイル開発手法に関するワークショップに参加した．これにより，短いサイクルでの計画・実行・評価を繰り返して，プロダクトを開発していくための手法を学んだ．

2.3　サービスの考案

本サービスは当初「ボーニクラフト」として考案されたサービスコンセプトが開発プロセスにおける検証とフィードバックを経て進化し，決定された最終的な成果物である．

ボーニクラフトの機能としてはユーザーオリジナルの棒二森屋を作成することである．ユーザーは理想的な建物の階層を決め，その建物に入れたい店舗を選択肢の中から選択する．これにより理想の棒二森屋が完成し，ユーザーが函館駅前に欲しい建物を把握することができる．

ボーニクラフトはデジタル上で完結させる予定だったが，開発のイメージを明確にするためアナログで紙の箱にほしい建物をかき，函館駅前の白地図上に配置するという体験をグループメンバーで行った．この体験を通して「実際に手を動かして建物を作る手軽さ」「話し合いをしながら配置を決める楽しさ」といったアナログ要素が生み出す利点に気が付いた．

このことからデジタル空間でのシミュレーションという側面よりも，「図工的な楽しさ」と「AR によるリアルな体験」を融合させることに重点が置かれ，サービスの作成方針を「スマホの中で完結するのではなく，箱を用いたリアルな体験を混ぜる」というように決定した．

2.3.1　関連研究調査

ミライスケッチキャッスル

「ミライスケッチキャッスル（ミライキャッスル）」は，子どもたちが「未来のまち」をテーマにデジタルで描いた絵を，大阪・夢洲の巨大プロジェクションマッピング「MEGA CANVAS」に夜空に浮かぶお城のように投影する参加型アートプロジェクトである [1]．このプロジェクトは万博で開催されており，アプリ形式での開発ではなくイベントとしての開発のモデルとして参考にした．

建替えデザインゲーム

志村秀明ら (2002) の目標空間イメージの編集によるまちづくり協議ツールの開発に関する研究：建替えデザインゲームによる景観形成手法の開発は住民の主体的なまちづくり活動による改善型の市街地更新を進めるためにワークショップを用いてイメージタイプの具体化とその効果を明らかにすることを目的とした研究である [2]．この研究では本サービスのワークショップ部分と似た形式のワークショップを実施しており，ワークショップの詳細設計のために参考にした．

おく ROOM

おく ROOM は室内に家具を配置していきレイアウトを作成し，家具を購入することができるアプリである [3]．3D 空間での操作感やカメラワーク，家具の回転方法などを本サービスのデジタル部分での操作の参考とした．

第 3 章　サービス「はこぺパッと」の企画と設計

3.1　サービス立案の経緯

「はこぺパッと」は，棒二森屋跡地に新しくできる建物にユーザーが興味を持ち，未来の函館駅前周辺について考えるきっかけを持ってもらうことを目的に立案された．棒二森屋周辺は現在，賑わいが低下し廃れた印象が強く，若者には棒二森屋に対する思い出や関心が薄いという課題認識があったため，サービスを通じてこの場所への関心を高めることを目指した．

サービス名は「パパッと」作れる手軽さ，そして「函館」「箱」「ペーパー（紙）」といった要素を組み合わせて「はこぺパッと」と決定された．

3.1.1　サービス案の決定とワークショップの実施

プロジェクトメンバーを対象としたプレワークショップを実施した結果，当初予定していたデジタル上での街作成ではなく，実際に「箱に絵を描く楽しさ」「話し合いながら配置を考える楽しさ」「理想の街を見上げる楽しさ」といった，箱を用いたリアルな体験が非常に楽しいというフィードバックが得られた．

この結果に基づき，楽しさの体験を重視する方針に決定し，「ペーパークラフト」と「AR 技術」を融合させた「はこぺパッと」を主要なサービスとして採用した．制作スタイルは，複雑なモデリングではなく，積み木やどうぶつの森のような手軽さを重視する方針が決定された．

3.2　サービスについて（概要と機能）

「はこぺパッと」は，ペーパークラフトと AR 技術を組み合わせ，ワークショップを通じて函館の街をデザインする未来のまちづくり体験ツールである．紙箱に描いた建物のデザインをスマートフォンで読み込み，AR 上の街にバーチャルな建物として出現させ，理想の街を作り散策できる．

本サービスの主な機能と体験内容は以下の通りである．

ユーザーは立方体や直方体の展開図に欲しい建物の絵を自由に描き，建物の外装をデザインする．建物に入れたい店舗を選択または自身で登録する機能が検討されており，これによりユーザーが本当に欲しいものを計測し，まちづくりに役立てる効果が期待される．

旧棒二森屋跡地以外では，作成した建物を手のひらサイズほどで AR 表示できる．現地では，現在の棒二森屋の建物にかぶせる形で AR 表示し，他のユーザーが作成した建物と共に街が作られていく様子を体験できる．街の中を歩く際，人間視点に加えて，ドローン視点や猫視点など，さまざまな視点に切り替えて街を体験できる．

第 4 章　手法・開発プロセス

本章では，サービス「はこぺパッと」の開発において採用した手法と，具体的な開発プロセスについて述べる．

開発は，アジャイル的なアプローチを取り入れ，機能ごとのプロトタイプ作成，検証，統合を繰り返す形で進められた．なお，開発段階ではプロジェクトコードネームとして「Bonicraft（ボーニクラフト）」という名称を使用している箇所がある．

4.1　チーム体制とプロジェクト運営

4.1.1　役割分担とリーダーシップ

本プロジェクトでは，開発フェーズに入る前のプロジェクト全体（15 名）での活動初期において，特定のリーダーを設けないという方針を採用した．これは，リーダーを固定することで特定の個人に業務や責任が集中し，プロジェクト運営が停滞するリスクを回避するためであり，指導教員である松原先生からの助言に基づいた決定であった．

15 名という大人数でのプロジェクト運営にあたり，全員が自由に発言する形式では議論の収束や進行が困難であると判断されたため，当初はファシリテーターと呼ばれる議長役を 1 週間ごとに交代で務める制度を導入した．しかし，この手法には課題が残った．ファシリテーターが実質的なリーダーとなり負担が依然として大きいことに加え，週ごとの交代に伴う情報伝達や引継ぎが円滑に行われず，意思決定のプロセスにおいて効率的な手法とは言えなかった．

これを受け，後期に入り「はこぺパッと」の開発が本格化する段階では，ファシリテーターが自然と固定化される運用へと移行した．形式上はリーダーを置く体制と類似しているが，明確なリーダーという職責を持たないという位置付けが，ファシリテーター（実質的な進行役）に心理的なゆとりをもたらした．この心理的な安全性が，結果としてチーム全体のスムーズな進行と，柔軟な開発体制の維持に寄与したと考えられる．

また，開発における役割分担についても同様の柔軟性を重視した．プロジェクト開始当初からエンジニアやデザイナーといった職能による固定的な役割定義は行わず，発生するタスクごとに，メンバー個人の「やりたいタスク（志向）」や「得意なタスク（適性）」を尊重して担当を振り分ける方針をとった．これにより，メンバーのモチベーションを維持しつつ，状況に応じた最適なリソース配分が可能となった．

このようなチーム体制のもと，以下の節で詳述するプロトタイプから実装に至る開発プロセスを推進した．

4.1.2　開発手法：アジャイルとウォーターフォールのハイブリッド的アプローチ

現在，ソフトウェア開発の手法に関してはアジャイルやウォーターフォールをはじめとする多種多様な体系が存在する．しかし，どの手法を採用すべきかはプロジェクトの状況によって変化するものであり，万能な正解は存在しないと考える．本プロジェクトにおいても，初期段階でアジャイル開発のワークショップに参加するなどして導入を模索したが，既存の枠組みをそのまま適用する

ことには限界を感じた．

アジャイルやウォーターフォールといった体系化された手法は，基本的にビジネスの現場を前提として設計されている．一方で，大学におけるプロジェクト学習は，業務とは異なり，投入した努力に対する金銭的な報酬や即時的な評価といったインセンティブ構造が希薄である．この報酬の不在は，開発のモチベーション維持において無視できない欠点となり得るため，ビジネスの論理で作られた手法を形式的に取り入れることは適切ではないと判断した．

そこで本プロジェクトでは，手法の形式よりも，メンバーのモチベーションを維持・向上させる要素を最優先事項とした．具体的には，「作っているものの面白さ」「自慢できるような成果物」そして「AI Agent による作業の効率化」の 3 点を重視した．面白さや成果物の質は結果論的な側面も強いが，AI Agent による効率化はあらゆるプロジェクトに適用可能な汎用的手法である．これについては後に（4.1.3）で詳述する．

結果として，本プロジェクトの進行は，特定の教条的な手法には依拠しないものの，実態としてはウォーターフォール的な要素とアジャイル的な要素を混合した，いわば「締切駆動開発」とも呼べる特殊なスタイルとなった．具体的な特徴は以下の通りである．

- ウォーターフォール的側面
  実現難易度の高い野心的な目標とゴールを設定し，最終発表という絶対的な締切に向けて全体像を構築した点．
- アジャイル的側面
  開発中に「ユーザー体験としての価値が低い」「効果が薄い」と判断した機能は即座に切り捨てる柔軟性を持たせた点．また，報酬（給料・残業代のようなインセンティブ）が発生しないチームであることを前提に，活動を授業時間内に限定して「定時」で終わらせるタイムボックス管理を徹底した点．さらに，2 時間の授業開始前にスプリントプランニングミーティングを実施し，短期間でのタスク調整を行った点などが挙げられる．
- 柔軟なリソース管理
  授業時間外での活動については，個々人の生活や別の活動を優先し，やる気のある時に進め，忙しい時は無理をしないという方針を採った．これにより，義務感による疲弊を防ぎ，能動的なコミットメントを引き出した．

これらの複合的なアプローチを採用したことにより，最終的に当初掲げた理想のゴールのすべてには到達しなかったものの，プロジェクトの成果として対外的に発表しうる品質のサービスの開発を達成することができたと考える．

4.1.3　生成 AI による効率化

前節で述べた柔軟なリソース管理により，時間を大量に投入することによる成果の向上という手法は採用できない．また，ツールの習熟自体が個々人のモチベーションに左右されてしまう．そこで本プロジェクトでは，生成 AI の活用をプロジェクトの効率化と質的向上のために積極的に推進した．コーディングに関する生成 AI の利用については 4.2 章で触れる．

企業などの機密保持が厳格な環境とは異なり，大学という比較的柔軟な情報利用が許容される環境の利点を活かし，以下のような場面で LLM（大規模言語モデル）等の利用を実践的に投入した．

- **DeepResearch による初期調査**
  棒二森屋についての情報の調査や既存サービスの調査の初期段階では，Gemini Deep

Research や ChatGPT Deep Research，Perplexity Deep Research を利用した．これにより，情報が全くない状態での情報のリサーチの初期速度を向上させた．

- **会議の分析と要約**:
  議事録や議論のメモを定期的に NotebookLM や ChatGPT，Gemini 等の LLM に投入し，論点の整理や要約を行わせた．

- **イメージの具現化**:
  言語化しにくい視覚的なイメージ（建物の外観や世界観など）を共有する際，画像生成 AI を用いて素早く具体例を作成し，チーム内の認識を統一した．

- **壁打ち相手としての利用**:
  アイデア出しに行き詰まった際や，思考を整理したい時に，チャットボットを対話相手（壁打ち相手）として利用した．これにより，停滞することなく議論を前に進めることができた．

このように，「思考や議論を加速させる AI ツール」を積極的に取り入れることで，限られたリソースの中で最大限の成果を生み出す体制を整えた．

4.1.4 運営ツールの選択

開発以外のプロジェクト運営，特にコミュニケーションやスケジュール管理，アイデア創出においては，導入コストと心理的障壁を最小限に抑えることを基本方針とした．

大学の授業や日常生活で馴染みのない複雑なツールは，操作の習熟（学習コスト）に時間を要するだけでなく，ツールを開くという行為自体に心理的なハードルを生じさせる．これは，前節で述べたモチベーション維持の観点からも避けるべきリスクであると判断した．

当初，プロジェクト管理ツールとして Notion の導入を検討した時期もあった．Notion は柔軟なデータベース機能やタスク管理，ドキュメント作成を一元化できる多機能ツールとして知られており，一見するとプロジェクト運営に最適に思えた．しかし，実際にチームに導入してみると，その多機能さゆえに操作方法を習得するまでの学習コストが想定以上に高いことが判明した．特に，プロジェクト管理ツールに不慣れなメンバーにとっては，「どこに何を書けばよいのか」「どのように情報を整理すればよいのか」といった基本的な部分でつまずきが生じ，ツールの習熟自体がプロジェクトの負担となる可能性が懸念された．

そのため，Jira や Backlog，あるいは多機能な Notion といった，一般的にアジャイル開発で用いられる高度なプロジェクト管理ツールの導入は見送り，以下の通りメンバーが直感的に扱えるツールを選定した．

1. **コミュニケーションと記録**
   意思決定やアイデア出しの記録には，物理的な紙媒体，または全員が使い慣れている Google Docs を採用した．特に Google Docs については，タブ機能を活用することで，1 つのドキュメント内に「会議記録」「メモ」「開発タスク管理」といった複数の情報ページを集約し，情報の一元管理を実現した．これにより，Notion のような専門ツールを導入せずとも，使い慣れた Google Docs だけで必要なプロジェクト管理機能を代替することができた．日常的な連絡や迅速な情報共有には，学生間で最も普及している LINE グループや，大学側から用意されていた Discord を活用した．これにより，連絡確認の遅延を防ぎ，カジュアルな雰囲気での活発な議論を促進した．
2. **スケジュール管理**
   進捗管理においても，厳格なガントチャートやチケット管理システムではなく，スマートフォンから手軽にアクセス・編集が可能なカレンダーアプリ「TimeTree」を採用した．機能がシンプルであるため，入力や確認の手間が少なく，メンバー全員が常に最新の予定を把握しやすい環境を構築した．

このように「管理コストのかかるツール」を徹底して排除することで，メンバーの負担を軽減し，本来の開発作業に集中できる環境を構築した．

4.2 技術選定と開発戦略

4.2.1 AI Agent 中心の開発体制と技術選定

本プロジェクトではアイデアの段階で，AR（拡張現実）を用いた 3D 表現が必須要件であったため，開発フレームワークの選定にあたり，ゲームエンジンである「Unity」と，Web 標準技術ベースの「Web (Three.js React)」の比較検討を行った．最終的な開発環境の仕様については 4.4

節で詳述するが，本節ではそこに至るまでの技術選定のプロセスと，AI Agent（自律型 AI）の導入を前提とした意思決定について述べる．

1. フレームワーク（3D エンジン）の選定　開発初期段階において，主要開発メンバー 3 名の技術スタックは Web 開発経験者と Unity 経験者が混在しており，どちらの技術を採用するかは大きな検討課題であった． Unity は AR/XR 開発においてスタンダードであり，高品質な 3D アプリケーションを構築できる利点がある．一方，Web ベース（Three.js）のアプローチは，専用アプリのインストールが不要で，ブラウザのみで動作するというアクセシビリティの高さが魅力である．開発チームは，4.1 節で後述するプロトタイプ作成を通じて双方の技術検証を行い，プロジェクトの規模と目指すべき開発スタイルに照らして検討を進めた．

2. Agent Coding への適合性と Web 技術の採用　技術選定においては，「AI Agent による開発効率化（Agent Coding)」を開発プロセスに導入した場合の適合性が，重要な検討要素の一つとなった．本プロジェクトでは，限られた期間とリソースで複数ファイルにまたがるコードを含む中規模のシステムを構築するにあたり，生成 AI によるコード記述支援や自動化を開発プロセスに取り入れることを試みた．

- **Unity**

Unity 開発は「Unity Editor」という GUI（グラフィカルユーザインタフェース）への依存度が高く，シーンの構築やオブジェクト設定においてマウス操作が必須となる場面が多い．開発時点（2025 年 8 月）において，AI Agent が GUI 操作を自律的に行うことは困難であり，MCP サーバー等を介した外部制御も発展途上の段階にあった．また，ビルドから実機テストまでのリードタイムが長く，高速な試行錯誤を阻害する要因となった．

- **Web 技術（Three.js　React）**

Web 開発は，構造（HTML），スタイル（CSS），ロジック（JavaScript/TypeScript）のすべてがテキストベースのコードで管理されるため，LLM（大規模言語モデル）との親和性が極めて高い．AI Agent に対して自然言語で指示を出し，コードを生成・修正させるサイクルを回しやすい環境である．さらに，Web 開発は「修正」から「プレビュー」までの反映が瞬時に行えるほか，ステージング環境の複製や URL によるチーム内共有が容易であり，フィードバックループを高速化できる利点がある．

以上の理由から，本プロジェクトでは AI Agent との協業に最適化された **Web（Three.js　React）** をメインの開発環境として採用することとした．

4.2.2　AI Agent 向け開発手法の選択

4.2.2.1 AI Agent 開発における手法の潮流

2025 年現在，AI Agent 技術の発展により，ソフトウェア開発の手法論は大きな転換期を迎えている．従来のアジャイルやウォーターフォールといった体系化された開発手法は，主に人間のエンジニアがコードを直接記述することを前提として設計されてきた．しかし，生成 AI（LLM）の実用化により，「自然言語での指示によって AI Agent が自律的にコードを生成・実装する」という新たな開発スタイルが急速に普及しつつある [4].

この潮流の起点として，2025 年 2 月に Andrej Karpathy が提唱した「Vibe Coding」がある

[4]．Vibe Coding は，厳密な仕様書を作成せず，「感覚」や「雰囲気」を重視した探索的なコーディング手法であり，AI Agent を用いた高速なプロトタイピングを可能にした．しかし，この手法は持続的な開発においては，コード品質の低下や設計の欠如といった課題を抱えていた [4]．これを受け，要件・設計・実装を明確に分離し，仕様を事前に定義する「仕様駆動開発（Spec-Driven Development, SDD）」へと発展した．SDD は，探索的なアイデア創出フェーズと，体系的な実装フェーズを両立させることで，AI 時代の開発における持続可能性を追求するものである．

さらに，AI Agent による開発手法は「Agentic Coding」とも呼ばれる．Agentic Coding では，エンジニアの役割が「コードを記述すること」から「AI に意図を正確に伝えること」へと転換し，自然言語のプロンプトを通じて AI が自律的に実装を進める．この過程では，探索・計画・実装のサイクルが高速に回転し，従来の手法では困難であった大規模な変更やリファクタリングも短時間で実行可能となる．また，AI Agent はターミナル操作やブラウザ操作を含む多様なタスクを自律的に遂行するため，開発者はより高次の設計や意思決定に注力できるようになった．

このような AI Agent 中心の開発手法は，従来の手法論の枠組みを超えた柔軟性を持つ一方で，プロンプト設計やコードベースの構造化といった新たなスキルセットを要求する．本プロジェクトでは，こうした時代的背景を踏まえ，複数の手法を組み合わせたハイブリッドなアプローチを採用した．

4.2.2.2 Agentic Coding：AI エージェントによる自律開発

Agentic Coding とは AI エージェントに計画立案から実装までを自律的に遂行させる開発手法である．従来のコード補完ツールが単発的な提案にとどまるのに対し，Agentic Coding では人間が高レベルの方向性を示し，AI エージェントが「探索・計画・実装・コミット」のサイクルを自律的に回す点が特徴である．開発者は技術選定や要件定義といった意思決定に注力し，詳細実装は AI エージェントに委ねることで，開発速度と品質の両立を図る．

主要ツールとして, GitHub Copilot と Claude Code, クラウド上で Agent 環境として, GitHub Copilot Coding Agent を利用した．これらのツールを効果的に活用するため, CLAUDE.md ファイルや copilot-instructions.md をコードベースに配置し，アーキテクチャ・コーディング規約・実装方針といったプロジェクトコンテキストを明示的に提供する手法を採用した．

本プロジェクトでは，Claude Code をマルチファイルリファクタリングに，GitHub Copilot を日常的な開発タスク処理に使い分けた．これにより，AI エージェントが一貫性のある実装を自律的に生成できる環境を構築した．各ツールの詳細な機能比較と基盤モデルの使い分けについては，4.5.1 節で述べる．

4.2.2.3 仕様駆動開発（Spec-Driven Development）

仕様駆動開発（Spec-Driven Development，以下 SDD）は，AI エージェントによる実装に先立って詳細な仕様書を自然言語で記述し，それをエージェントが解釈して実装を行う開発手法である．この手法は，Andrej Karpathy が提唱した「Vibe Coding」（直感的なコーディング）から進化したアプローチとして，2025 年以降の AI エージェント開発における重要な潮流となっている．

SDD の基本的なワークフローは，Requirements（要求定義）→ Design（設計仕様）→ Tasks（実装タスク）という 3 段階から構成される．まず要求を自然言語で明確化し，次にそれを実装可能な設計仕様へと詳細化し，最後にエージェントが実行可能なタスクへと分解する．このプロセスにより，AI エージェントに「何を作るべきか」を明確に伝えることができる．主要なツールとしては，Amazon Kiro[5]，GitHub Spec Kit[6] などが知られており，いずれも仕様書の構造化とエー

ジェントへの伝達を支援する機能を持つ.

SDD の主なメリットは，仕様の明確化による実装の手戻り防止，複数エージェント間での仕様共有による協調開発の実現，そして人間と AI の間のコミュニケーションギャップの縮小にある．一方で，詳細な仕様書を作成するコストが発生することや，プロジェクト初期には有効だが中期以降には仕様書の維持管理が硬直的になる可能性があるという課題も存在する．

本プロジェクトでは，Phase 1（探索・仕様形成フェーズ）において Kiro を用いた SDD を積極的に活用した．コードベースが未成熟で文脈が不足している段階では，詳細な仕様書がエージェントへの「橋渡し」として機能し，実装の方向性を効果的に制御できた．しかし Phase 2（エージェント主導開発フェーズ）では，既存コードベース自体が十分なコンテキストを提供するようになったため，SDD の比重を意図的に縮小し，ラフな指示による即実装スタイルへと移行した．この判断により，開発速度を維持しながら仕様作成コストを削減することができた．SDD の詳細な適用事例および各フェーズにおける具体的なワークフローについては，4.5.2 節を参照されたい.

4.2.2.4 API 契約駆動開発（Contract-First Development）

API 契約駆動開発（Contract-First Development）は，API 仕様を実装に先立って定義し，その仕様を「契約」として各チームが並行開発を進める手法である．従来の Code-First アプローチではバックエンドとフロントエンドの開発が逐次的になりやすく，実装後の仕様変更が手戻りを生む課題があった．これに対し，Contract-First では最初に OpenAPI/Swagger 仕様などの形式で API の入出力を確定し，フロントエンド・バックエンド双方がその契約に基づいて独立に実装を進める．

特に TypeScript を採用した開発環境では，型付け機能を活用したデータ契約が有効である．型定義ファイル（types/）を API 契約として機能させることで，コンパイル時に型の不整合を検出でき，ランタイムエラーを未然に防ぐことができる．本プロジェクトでは，フロントエンド（React）の types/ディレクトリに詳細なコメント付きの型定義を集約し，これをバックエンド（FastAPI）設計時の「仕様書」として活用した.

複数リポジトリにまたがる開発では，AI Agent が他リポジトリのコンテキストを持たないという課題がある．この課題に対し，openapi.json をリポジトリ間連携の「API 契約」として利用するアプローチが効果的であった．FastAPI が自動生成する OpenAPI 仕様をフロントエンド側の Agent セッションで参照することで，バックエンドの実装詳細を知らずとも正確な API クライアントを生成できた．このフロントエンド優先・API 契約駆動の開発フローにより，リポジトリ間の疎結合を維持しつつ，型安全性を担保した並行開発が可能となった．詳細なプロセスと各ステップの実装については，4.5.5 節「複数リポジトリにまたがる開発サイクル」で述べる.

4.2.2.5 本プロジェクトにおけるハイブリッドアプローチ

本プロジェクトでは，Agentic Coding，SDD，API 契約駆動開発，という 3 つの手法を組み合わせたハイブリッドアプローチを採用した．この選択は，限られたリソースの中で複数プラットフォーム（Web，iOS，VR）に対応した中規模なシステムを短期間で開発するという現実的な制約に対応するためである.

開発フェーズによって手法の比重を動的に変化させる戦略をとった．Phase 0（環境構築）では，Vite Template や Kiro による環境定義ファイルを活用し，CI/CD 環境を自動化した．Phase 1（探索・仕様形成）では，SDD を中心に据え，Kiro の「Plan Mode」や Claude Code で仕様を言語化し，AI エージェントへの明確な指示として機能させた．同時に，TypeScript の型定義を「契

約」として整備し，複数リポジトリ間の連携基盤を構築した．Phase 2（エージェント主導）では，成熟したコードベース自体がコンテキストとなるため，SDD の比重を下げ，ラフな指示による即実装スタイルへと移行した．GitHub Copilot や Claude Code が自律的にコード生成を行い，開発速度を最大化した．

このハイブリッドアプローチの利点は，柔軟性，開発速度，品質維持，そしてチーム内の技術差の吸収という 4 点にある．フェーズごとに最適な手法を選択することで，仕様の不確実性が高い初期段階では手戻りを防ぎ，中期以降は実装速度を重視する柔軟な体制を実現した．AI エージェントの特性を最大限活用することで，少人数チームでありながら高速な開発サイクルを回すことができた．また，定期的なリファクタリングフェーズを設けることで，高速開発による技術的負債の蓄積を防ぎ，コードの保守性を維持した．

さらに重要な利点として，チーム内にプログラミング経験や知識の差が存在する場合でも，AI Agent を活用することで，その差を吸収できた点が挙げられる．従来の開発手法では，新しいプログラミング言語やフレームワークを習得するために相当な学習時間が必要であり，これがプロジェクト参画の障壁となっていた．しかし，AI Agent 中心の開発体制では，ある程度のプログラミングの基礎知識とデバッグ能力（エラーメッセージの読解やログの確認など）さえあれば，特定言語の詳細な文法や構文を習得せずとも実装作業に取り組むことが可能であった．メンバーは自然言語で意図を伝え，AI Agent が具体的なコードを生成するため，言語学習のオーバーヘッドを大幅に削減でき，開発への参加障壁を下げることができた．

成果として，WebAR，iOS ネイティブアプリ，MetaQuest 対応という 3 つのプラットフォームへの展開を，限られたリソースの中で達成することができた．

これらの手法を具体的にどのように実装したかは，4.5 節で詳述する．

手法	特徴	適用フェーズ	主要ツール
SDD	仕様の明確化、実装の手戻り防止	Phase 1（初期）	Kiro, Spec Kit
API契約駆動	リポジトリ間連携、並行開発	Phase 1-2	TypeScript, OpenAPI
Agentic Coding	自律的実装、高速開発	Phase 2（中期以降）	Claude Code, Copilot

4.2.3 WebXR における課題とハイブリッド構成への移行

Web 技術の採用により機能実装は順調に進んだものの，AR 機能の設計・実装段階において，iOS（iPhone/iPad）の WebXR 対応状況に起因する重大な課題に直面した． Android 端末などでは WebXR を用いたブラウザ上での AR 体験が可能であったが，iOS の Safari ブラウザは WebXR に完全対応しておらず，Web 技術のみでは意図した AR 体験を提供できないことが判明した．開発途中で Unity への全面移行も再検討したが，前述の開発効率の観点から，実装コストが過大であると判断した．最終的な解決策として，クロスプラットフォーム対応を前提としつつ，OS ごとの特性に合わせたハイブリッドな構成を採用した．

- **基本構成:** アプリケーションのコアロジックおよび 3D 描画処理は Web（React Three.js）で実装する．
- **iOS 対応:** WebXR が動作しない iOS 環境に対してのみ，Apple 純正の AR フレームワークである ARKit と Swift を用い，Web 側のコンテンツを表示・連動させるための専用プレビューアプリ（ネイティブアプリ）を開発した．

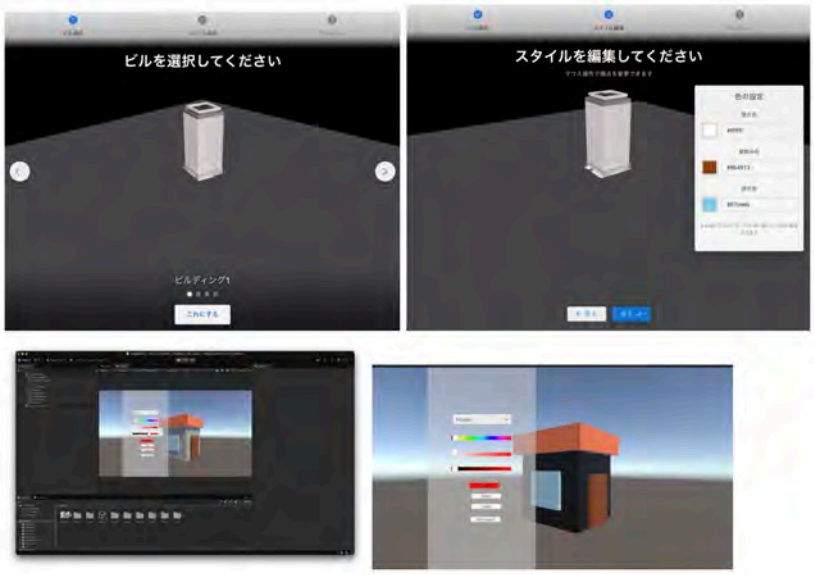
これにより，AI Agent を活用した Web ベースの高速開発体制を維持しつつ，デバイス依存の制約をネイティブ技術で補完する開発環境を確立した．また，WebXR(Three.js) を採用した利点として，MetaQuest へ対応がとても簡単で，当初はスマホのみでの実装を検討していたが，最終的に VR ゴーグルでの体験も同時に実装することができた．

4.3 プロトタイピングと技術検証

本プロジェクトでは，アイデアの意見出しと並行して，技術的な実現可能性を確認するためのプロトタイピングを早期から実施した．特に「紙に描いた展開図を 3D 化する」というコア機能と，AR 表示には，実現可能かの技術的な不確実性が存在したため，機能ごとに分割したプロトタイプを作成し，段階的に検証を進める手法をとった．

4.3.1 機能別プロトタイプの検証

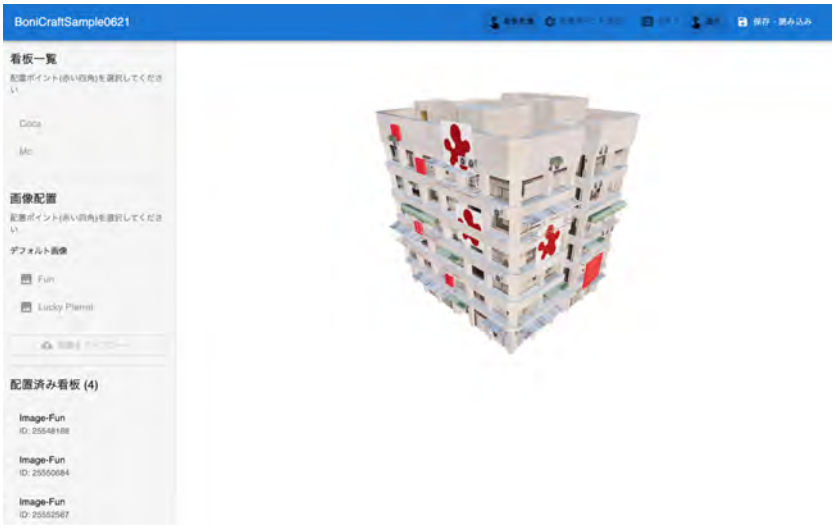
1. **建物選択と 3D プレビューの検証**



Web 技術 (Three.js) と Unity のどちらを採用すべきかを判断するため，双方のフレームワークで同様の機能を実装し，比較検討を行った．

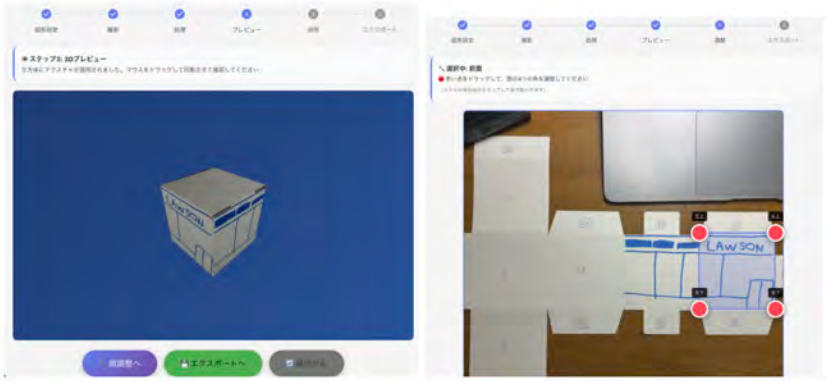
検証結果: 実装コストと Web 上での動作軽快性を比較した結果，Three.js の採用が適切であるとの判断に至った．

2. **建物編集機能の検証**



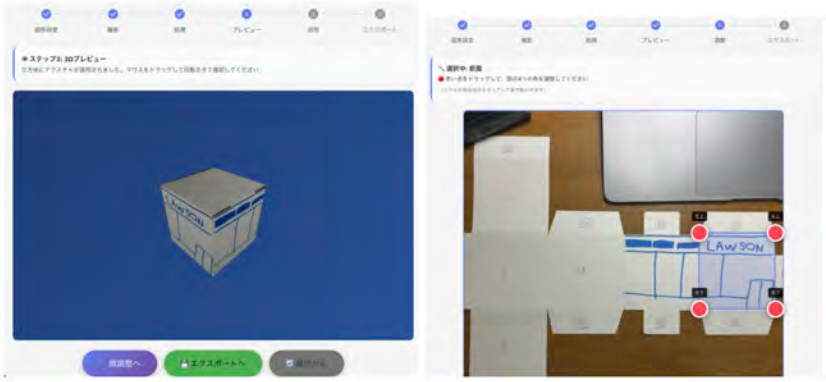
既存の建物モデルに対し，看板などのパーツを自由に貼り付けるカスタマイズ機能のプロトタイプを作成した．

検証結果: 単純なオブジェクト配置のみではエンターテインメント性や創造性が不十分であり，ユーザー体験としての魅力に欠けることが判明した．これにより，より自由度の高い「ブロック積み上げ方式」への転換が必要であるとの知見を得た．



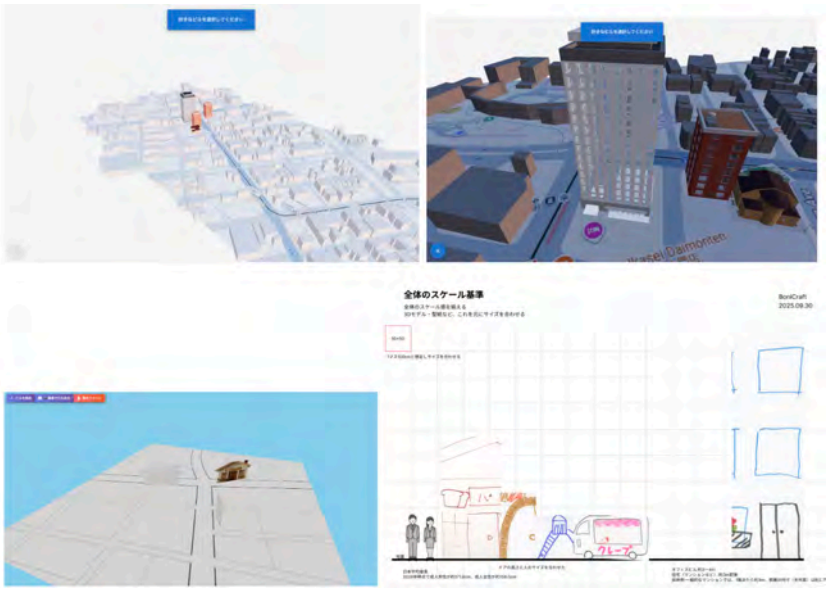
3. **カメラスキャン機能，画像編集・変換処理の検証**

本サービスの核となる「手描きの展開図を 3D ブロックに変換する機能」の実現性を検証した．具体的には，スマートフォンのカメラで撮影した画像から，OpenCV 等のライブラリを用いずにブラウザ上の JavaScript のみで射影変換（ホモグラフィ変換）を行い，立方体のテクスチャとして正しく切り出せるかをテストした．

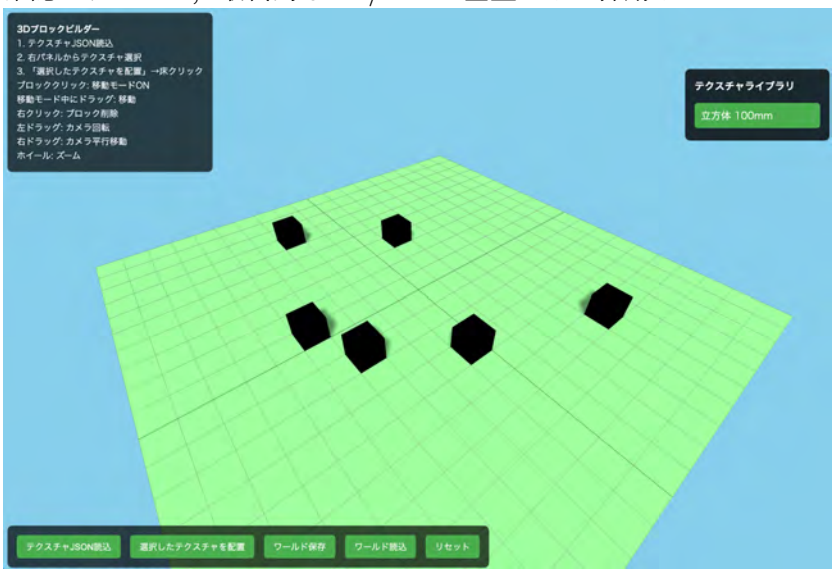


4. **ワールド編集プロトタイプ**

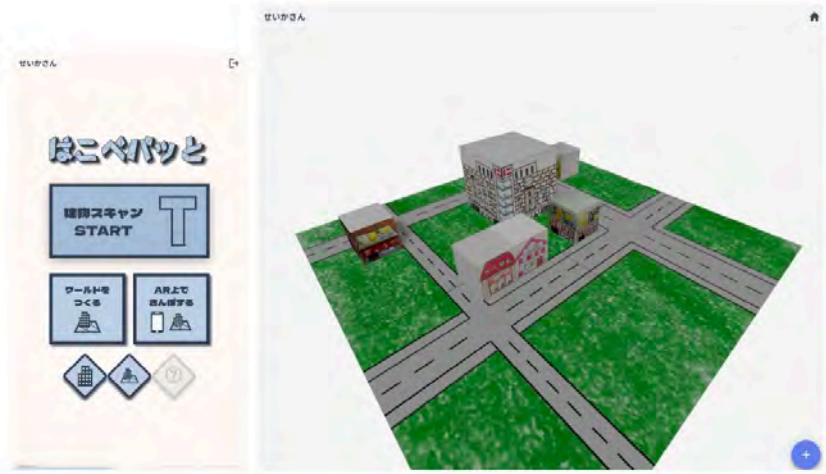
街づくりの体験を検証するため，様々な手法を検討した．完成されたモデルを配置する案や，写真の切り抜きを配置する「飛び出す絵本」風のアプローチなどのプロトタイプを作成した．



5. **グリッドベースのワールド作成検証**
   物理的なワークショップに近い操作感をデジタル上で再現することを目的としたプロトタイプ．グリッド状のフィールドを作成し，マウスやタッチ操作で直感的にブロックを積み上げられるかを確認した．
   検証結果: この操作体系が最も直感的であり，ユーザーがスケール感を把握しやすいことが確認されたため，最終的な UI/UX の基盤として採用した．



6. **統合プロトタイプ（Bonicraft）**
   上記で検証した各機能を統合し，ログインから建物作成，AR 表示までの一連の流れを動作させるための統合環境を構築した．



これらのプロトタイプ開発を通じて，WebAR 技術（WebXR）を用いた実装の実現可能性を確認し，Safari 等のブラウザにおける制約への対策を講じる基礎データを収集した．

4.4 デザインとユーザー体験

本節では，ユーザーが直感的に操作でき，かつ「紙とデジタルの融合」というコンセプトを体感できるようなデザインとユーザー体験（UX）の設計プロセスについて述べる．

4.4.1 画面遷移と UX 設計

サービスのユーザー体験（UX）を定義するため，ワークショップでの利用シーンを想定した画面遷移図を作成した．Figma を使用し，ユーザーが迷わずに操作できるよう，以下のフローを中心に設計を行った．

1. **ロード・ホーム画面**：ロゴと「建物スキャン」「みんなの建物を見る」などの主要アクションへの導線配置．
2. **建物スキャン画面**：展開図を撮影するためのガイド表示と，撮影後のプレビュー確認．
3. **建物詳細入力画面**：建物名やカテゴリー（お店の種類など）の登録．
4. **AR 表示・編集画面**：作成した建物を AR 空間に配置し，ドローン視点・人間視点・猫視点などで閲覧する機能．
5. **ギャラリー画面**：自分や他者が作成した建物を一覧表示し，「いいね」などの評価を行う機能．

設計段階では，当初の多機能なメニュー構成から，ワークショップでの体験を重視したシンプルな構成へと修正を重ねた．

4.4.2 UI デザインと世界観の構築

画面デザイン（UI）は，子供から大人まで親しみやすい「ペーパークラフト」の世界観を意識して制作された．

- **デザインコンセプト**：「紙・本の世界」をテイストとし，デジタルでありながらアナログな工作の楽しさを損なわないデザインを目指した．
- **配色・アイコン**：視認性を高めるため，コントラストを意識した配色と，直感的に機能を理

解できるアイコン（ドローン，人，猫のアイコン等）を採用した．

- **ロゴデザイン**：「はこぺパッと」の名称決定に伴い，「箱」「ペーパー」「AR」「パパッと作れる」という要素を反映したロゴを作成し，親しみやすさを強調した．

4.5 最終的な開発体制

本節では，4.2 節で述べた技術選定を踏まえ，最終的に確立した開発体制の詳細について述べる．AI Agent を中心とした開発において，ツールの選択，プロンプト設計，コードベースの構造化がいかに重要であったかを整理する．

4.5.1 AI Agent ツールと基盤モデルの選択

本プロジェクトでは，複数の AI Agent ツールと基盤モデルを用途に応じて使い分ける体制を構築した．

- **Coding Agent（コード生成・実装）**
  - **GitHub Copilot / GitHub Copilot Coding Agent**: IDE（VS Code）上でのリアルタイム補完や，PR ベースの自律的な実装に使用した．特に GitHub Copilot Coding Agent は，Issue を与えるとバックグラウンドで実装を進め，PR を自動生成する機能を活用した．
  - **Claude Code**: ターミナル上で動作する Agent として，ファイル横断的なリファクタリングや，複雑なロジックの実装に活用した．長いコンテキストを保持できる特性を活かし，大規模な変更タスクに適している．
  - **Kiro（AWS）**: 仕様駆動開発（SDD）に特化した Agent として，プロジェクト初期の仕様定義とタスク分解に活用した．
- **AI Assist・Prototyping（設計支援・プロトタイプ）**
  - **Claude Artifacts / Google Gemini Canvas**: 会議中のアイデアを即座にプロトタイプ化する際に使用した．特に React コンポーネントの雛形作成や，UI のモックアップ生成に有効であった．
  - **ChatGPT / Gemini**: 技術調査，ドキュメント作成，壁打ち相手としての対話に使用した．
- **基盤モデルの特性と使い分け**

  上記ツールの背後にある基盤モデル（GPT，Claude，Gemini 等）は，それぞれ得意分野が異なる．本プロジェクトでは，コード生成には Claude を，長文の分析や要約・リサーチには Gemini を，アイデア出しには ChatGPT を，といった形で特性に応じた使い分けを行った．開発当初（8 月頃）は，Claude 3.5 Sonnet を利用していた．当時，SWE-bench（ソフトウェアエンジニアリング能力のベンチマーク）で高スコアを記録し，Agent 主導の開発に適していたのが Claude であったためである．その後，Claude 4 Sonnet 等のリリースに伴い，さらに自律作業の質が向上したため，開発終盤まで一貫して Claude を採用した．チーム内でも出力品質を統一するため，同一モデルの利用を徹底した．

4.5.2 仕様駆動開発（SDD）とコードベースの構造化

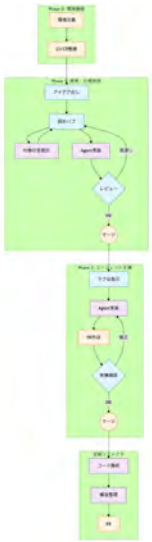
AI Agent を効果的に活用するためには，Agent が理解しやすい形で仕様を言語化し，コードベースを構造化することが重要である．本プロジェクトでは，開発フェーズに応じてこのアプローチを柔軟に調整した．

仕様駆動開発（Spec-Driven Development）の導入と段階的縮小

プロジェクト初期（Phase 1: 探索・仕様形成フェーズ）では，Kiro の「Plan Mode」や Claude Code を用いた**仕様駆動開発（SDD）**を採用した．これは 4.2.2.3 項で述べた通り，実装に先立って詳細な仕様書を自然言語で記述し，AI にその仕様を解釈させて実装を行わせる手法である．

- **初期フェーズでの SDD 活用**:
  コードベースが未成熟で，Agent がプロジェクトの文脈を十分に理解できない段階では，詳細な仕様書が不可欠であった．Kiro 等が生成する Tasks（実装タスク）リストにより，大規模な作業を適切に分割し，Agent 主導の開発を安全に進めることができた．
- **中期以降の SDD 縮小**:
  一方，コードベースがある程度成熟した中期以降（Phase 2: エージェント主導開発フェーズ）では，SDD の比重を意図的に下げた．これは，既存のコードベース自体が十分なコンテキスト（文脈）を提供するようになり，Agent が既存の構造や規則を自律的に参照できるようになったためである．仕様書の作成と維持には相応のコストがかかるため，頻繁な変更が発生するフェーズではボトルネックになり得る．そのため，この段階では「○○画面にこの機能を追加したい」といったラフな指示による即実装スタイルへと移行し，開発速度を優先した．

以下の図は，各開発フェーズにおける人間（青），AI Agent（紫），インフラ（橙）の役割分担と，SDD からラフ指示への移行を示している．



コードベースの構造化

Agent が効率的にコードを理解・生成できるよう，以下の構造化を意識した．

- **明確なディレクトリ構成**: 機能単位でディレクトリを分割し，Agent が関連ファイルを特定

しやすい構造とした．

- **一貫した命名規則**: コンポーネント名，関数名，変数名に一貫性を持たせ，Agent が命名パターンを学習しやすくした．
- **README / 設計ドキュメントの配置**: 主要ディレクトリに README を配置し，Agent がプロジェクト構造を把握する際の起点とした．

4.5.3 技術スタックとデータモデル

本サービスは，Web ブラウザで動作するクロスプラットフォームなアプリケーションとして構築された．

- **フロントエンド**: React 19 React Router 7 / Three.js React Three Fiber / TypeScript / Vite
- **バックエンド**: Python (FastAPI) / Google Cloud Platform (Cloud Run)
- **データベース・ストレージ**: Firestore / Google Cloud Storage
- **ホスティング**: Firebase Hosting （PR ごとにプレビュー環境を自動生成）
- **CI/CD**: GitHub Actions

データモデルについては，3D 空間を構成するブロック（建物）のデータを JSON 形式で定義した．各ブロックは位置座標，回転情報，各面のテクスチャ画像 URL，スケール情報を持つ構造とした．API は RESTful 形式で，日本語の「合言葉」を用いた簡易認証（パスワードレス認証）を採用した．

4.5.4 エージェント主導の開発サイクル

本プロジェクトでは，開発フェーズを「環境構築」「実装初期 (探索・仕様形成)」「実装中期 (エージェント主導)」「定期リファクタリング」の 4 段階に区分し，コードベースの成熟度に応じて人間と AI エージェントの役割を動的に変化させる開発フローを採用した．このプロセスは，少人数のチームで大規模な実装を短期間で遂行するための戦略的アプローチである．以下に各フェーズの詳細を述べる．

(1) 環境構築フェーズ：自動化による基盤整備

開発の初動段階（Phase 0）では，Vite Template および Kiro による環境定義ファイルを活用し，フロントエンド（React TypeScript）およびバックエンド（FastAPI）の環境を迅速に構築した．並行して，GitHub Actions を用いた CI/CD 環境を整備し，Pull Request（PR）が作成されるたびにプレビュー環境が自動生成される仕組みを導入した．これにより，開発者以外のメンバーも即座に Web ブラウザ上で動作確認が可能となり，フィードバックループの大幅な短縮を実現した．

(2) 実装初期：探索・仕様形成フェーズ

プロジェクト初期（Phase 1）は，WebAR 技術の検証やデータモデルの確定が必要なため，人間が中心となって設計を行い，AI がそれを補佐する体制をとった．

1˙タスク分解と仕様の言語化: 会議で創出されたアイデアを人間がタスク単位に分解した後，Kiro や Claude Code の「Plan Mode」を活用して，実装に必要な仕様を「短く決定的な文章」として言語化した．このプロセスにより，曖昧な要件を AI が解釈可能な明確なプロンプトへと変換

し，実装の手戻りを防いだ．

2˙Coding Agent による高速プロトタイプ: 定義された仕様に基づき，クラウド上の GitHub Copilot Agent 等がバックグラウンドで実装を行った．

3˙方向性確認レビュー: この段階でのレビューは，コードの細部よりも「仕様通りに動作するか」「技術的に実現可能か」という方向性の確認に重点を置いた．問題があれば設計プロセスに戻り，仕様を再定義するサイクルを回した．

(3) 実装中期：エージェント主導開発フェーズ

コードベースがある程度成熟した中期以降（Phase 2）は，詳細な仕様書を作成するコストを削減し，開発速度を最大化するアジャイルな手法へと移行した．

1˙ラフな指示による即実装: 「○○の機能をこの画面に追加したい」「ここの挙動を修正したい」といったラフなアイデアや改善点を，直接 Agent に投入するスタイルを採用した．既存のコードベースが十分なコンテキストを持っているため，Agent は意図を汲み取り，即座に実装を行うことが可能となった．

2˙実機レビュー中心の検証: 生成された PR に対しては，コードレビューよりも実機（スマートフォンやタブレット）での動作検証を優先した．特に AR 機能や UI の操作感はコードからは読み取れないため，実際に触ってデバッグを行い，修正があれば再度 Agent に指示を出すサイクルを回した．

(4) 定期リファクタリング：技術的負債の解消

エージェント主導の高速開発は，副作用としてコードの肥大化や構造の無秩序化を招きやすい．これを防ぐため，定期的にリファクタリング専用のタスクを Agent に投入した（Refactor フェーズ）．

1˙コードベース全体解析と構造整理: Agent にプロジェクト全体を解析させ，重複コードの排除，巨大コンポーネントの分割，ファイル構造の整理などを自律的に行わせた．

2˙大規模変更の適用: 人間が行うには心理的ハードルが高い大規模なリファクタリングも，AI であれば人間の介入なくで遂行可能である．レビューが負担になるという欠点はあるが，これにより，機能実装の開発速度を落とすことなく，コードの保守性を維持した．

以上のサイクルにより，「はこぺパッと」の開発は，Web と iOS のハイブリッド構成や複雑な AR 機能を持ちながらも，プロトタイピングから本番実装までを効率的に完遂することができた．

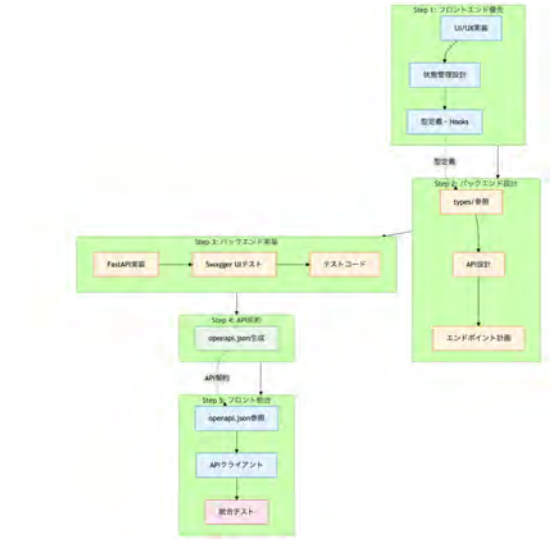
4.5.5 複数リポジトリにまたがる開発サイクル

本プロジェクトでは，フロントエンド（React），バックエンド（FastAPI），iOS ネイティブアプリ（Swift）を**別々のリポジトリ**で管理する構成を採用した．この構成において AI Agent を効果的に活用するため，リポジトリ間の連携を意識した開発サイクルを確立した．

```
├── bonicraft/          # フロントエンド（React + TypeScript）
│   ├── src/
│   │   ├── types/      # 型定義（API連携の橋渡し）
│   │   ├── hooks/      # 状態管理（jotai/atom）
│   │   └── ...
│   └── plan/           # 設計ドキュメント
│
├── bonicraft-api/      # バックエンド（FastAPI + Python）
│   ├── app/            # APIエンドポイント
│   ├── tests/          # テストコード
│   └── plan/           # 設計ドキュメント
│
└── bonicraft-ios/      # iOSネイティブ（Swift + ARKit）
```

フロントエンド優先・API 契約駆動開発（Contract-First API Development）

複数リポジトリ間で AI Agent を活用する際の課題は，**Agent が他リポジトリのコンテキストを持たない**ことである．これを解決するため，API 契約駆動開発（Contract-First API Development）の考え方を取り入れた「フロントエンド優先・API 契約駆動」の開発フローを採用した．



Step 1: フロントエンド優先実装

バックエンドが存在しない状態でも動作するよう，フロントエンドから実装を開始した．この段階で重要なのは，状態管理（jotai/atom）を API 連携を意識した形式で設計することである．

- 状態データを JSON 互換の構造で設計し，将来の API 通信を容易にした．
- ‘types/‘ディレクトリに型定義を集約し，各プロパティに詳細な JSDoc コメントを付与するよう Agent に指示した．この型定義とコメントが，後のバックエンド開発における「仕様書」として機能する．
- ‘hooks/‘ディレクトリに状態管理ロジックを集約した．API 呼び出しを行う箇所（将来的に fetch を使う箇所）を明確に分離し，バックエンド実装時にどのようなエンドポイントが必要かを Agent が把握しやすい構造とした．

Step 2: バックエンド Agent でフロントのコードを参照

バックエンド開発時には，別リポジトリであるフロントエンドの ‘types/‘や ‘hooks/‘ディレクトリを Agent のコンテキストに含めた．これにより，Agent はフロントエンドがどのようなデータ構造を期待しているかを理解し，適切な API エンドポイントを計画できた．

- Claude Code や Kiro の「Plan Mode」を用いて，フロントの型定義から逆算して API 仕様を設計した．
- 必要なエンドポイント，リクエスト/レスポンス形式を明確化した後に実装に移行した．

Step 3: バックエンド実装とテスト

FastAPI による実装後，Swagger UI（自動生成される API ドキュメント）を用いて実際にリクエストを送信し，レスポンスを検証した．この段階で API テストコードも作成し，リグレッション防止を図った．

Step 4: openapi.json の抽出

FastAPI が自動生成する ‘/openapi.json‘エンドポイントから，完成した API 仕様を OpenAPI 形式で抽出した．このファイルが，フロントエンドとバックエンド間の「API 契約」として機能する．

Step 5: フロントエンドで openapi.json を参照して実装

フロントエンド側の Agent セッションで，抽出した ‘openapi.json‘をコンテキストに含めた．これにより，Agent はバックエンドの実装詳細を知らなくても，正確な API クライアントを生成できた．

- 多くの場合，‘openapi.json‘のみで十分な情報が得られた．
- 複雑なケースでは, バックエンドの該当実装箇所もコンテキストに追加することで解決した．

このフローの利点

1. **リポジトリ間の疎結合を維持**: 各リポジトリは独立して開発可能．
2. **型定義と API 仕様が「API 契約」として機能**: Agent が他リポジトリのコンテキストを持たない問題を解決．
3. **フロントエンド優先による UX 重視**: ユーザー体験を先に確定し，それに合わせてバックエンドを設計できる．

第 5 章　サービス検証と改善

5.1　デモ会

開発の初期段階において，機能実装の進捗を共有し，チームメンバーやプロジェクト関係者からフィードバックを得る場として定期デモ会を実施した．この定期デモ会は 2 週間に一回のペースで実施された．

初回のデモ会では，5 人程度のチームで理想の「函館の街」をデザインし，手描きの建物をスキャンして AR で再現する体験型 AR アプリ「はこぺパッと」のコンセプトとプロトタイプを紹介した．

第二回デモ会では，プロジェクトメンバーを対象にユーザーが立方体の展開図に建物を描き，それをスキャンして建物名やカテゴリーを登録した後，「マイギャラリー」から AR 表示を体験する一連の流れを紹介した．AR 表示においては，ドローン視点や人視点への切り替え，他の作品の鑑賞，「いいね」機能による評価機能を披露した．

デモ会で得られたフィードバックから，ワークショップの目的として「街づくりにつながるか」という観点をより明確に示す必要があることが分かった．また，AR 体験が参加者のアイデアに変化をもたらすのか，AR 体験後スムーズに議論に戻れるのか，といった示唆が得られた．その結果をもとに，ユーザー間の会話を重視するためコメント機能の廃止，建物への「いいね」機能のみを実装することが決定された．

5.2　プレワークショップ

サービスの体験フローを検証し，ワークショップの質および操作性を向上させるため，未来大生を対象としたプレワークショップを二回実施した．これらの検証は，単なる技術的な動作確認にとどまらず，参加者が「理想の街づくり」に没頭できるか，またアナログとデジタルの融合がスムーズに行われるかといった体験価値の観点から詳細な分析を行うことを目的とした．

実施にあたり，参加者が意見を出しやすい環境づくりや，ファシリテーターの適切な介入方法についても検討を重ねた．その結果，「はこぺパッと」をより実用的で楽しいサービスにするための具体的な課題が浮き彫りとなった．

プレワークショップを通じて，主に以下の課題が抽出された．

- ワークショップの段取りが細かく，ファシリテーターの負担が大きい
- 今の函館駅周辺がよく分からなく，何を書けばいいのかわからない
- 「絵が苦手」という理由から創作のハードルが高い

これらの課題に対し，以下の改善を講じた．

- ワークショップの手順を分かりやすくまとめたスライドを作成し，説明を簡略化した．

- ワークショップの初めに Google Earth などを使用して現在の状況を確認する時間を設けた．また，台紙上に何も描かれていない状態を避けるため，あらかじめ描いた建物を配置しておくことで，参加者が描き始めるハードルを下げた．

- 他のユーザーが描いた建物を再利用できる「ギャラリー機能」を追加実装し，街づくりを進める際の創作のハードルを低減した．

第 6 章　成果

6.1　最終成果発表会

6.1.1　発表形式

最終成果発表会では 10 分で説明を行った．質疑応答は 10 分の説明の後に 5 分間で行われた．その際，発表の全体の場で質疑応答を行うのではなく，質問がある人は個人的に話すなどというような形をとった．そのため，質問がない人はアプリのデモンストレーションをやってもらうなど，VR ゴーグルの体験などをしてもらった．発表では，スライドをもとに背景，現状の課題，サービス概要，ワークショップのデモンストレーション，提供する価値，今後の展望を説明した．ワークショップのデモンストレーションでは，実際に箱と台紙を用いて実演形式で行った．デジタルに取り込む工程の後は動画を用いて説明した．動画を用いることで，分かりやすく伝えることができた．また，画面共有をする時よりも電波障害を避けることができ，スムーズに説明をすることができた．

6.1.2　発表技術の評価と反省

発表技術に関しては，高評価とされた意見として次が挙げられた．

・声量や抑揚，話すテンポ，間の取り方が適切で，聞き取りやすく安心して内容を理解できる発表であった．

・デモや動画を用いて実際の動作を示すことで，サービスの内容やできることが直感的に伝わってきた．

・スライドの構成と説明の対応関係が明確で，アナログ要素と AR を組み合わせた見せ方により，サービスの独自性が強く印象付けられた

低評価とされた意見は特に見受けられなかった．平均点は 9.2 点であった．最終発表では，実際のデモを会場に設置し，参加型の体験を交えたワークショップや動画を取り入れることで，AR という技術的に理解が難しくなりがちな内容を具体的に示すことを意識した．その結果，聞き手がサービスの利用イメージを持ちやすくなり，理解の促進につながったと考えられる．

6.1.3　発表内容の評価と反省

発表内容に関しては，高評価とされた意見として次が挙げられた．

・紙（クラフト）と AR を融合させたアイデアが独創的で，実空間と仮想空間の両方を活用した体験が具体的に伝わってきた．

・制作の背景や目的，ワークショップの流れが整理されており，対象に何を行うサービスなのかが分かりやすかった．

・実物や実機を用いた説明により，サービスの利用イメージや技術的な完成度を直感的に理解できる発表内容であった．

低評価とされた意見として次が挙げられた．

・応用できる可能性について示されている一方で，特に注力した利用シーンを一つにしぼって強

調すると，より印象に残る内容になると思う．

・技術的な工夫や強みについて，開発上の苦労や工夫をもう一段階具体的に示すことで，プロダクトの価値がさらに伝わる可能性がある．

・今後の発展や改善の方向性を簡潔にまとめることで，プロジェクトの将来性をより明確に示せると考えられる．

平均点は 9.3 点で，内容のわかりやすさとアイデア性の両面で高評価を得た．また，紙（クラフト）と AR 技術を融合させるアイデアについても高い評価が得られた．実空間へのブロック投影と，クラフト空間の散策という両方の体験をサポートしている点が独自性として評価され，まちづくりや再開発に限らず，他のワークショップへの応用可能性についても言及されていた．さらに，実物や実機を用いた説明，ワークショップの流れを参加型で示した点により，サービスの利用イメージが具体的に伝わったという意見も多かった．これにより，プロダクトの完成度や技術力の高さだけなく，「実際に使われる場面」を創造しやすい発表となった．総合的に見て，本発表はアイデアの新規性，構成の分かりやすさ，実演を通じた具体性を兼ね備えており，プロジェクト学習における成果を十分に示す内容であったといえる．

6.2　今後の展望

本プロジェクトでは，これまでに実施したプレワークショップで得られた知見をもとに改善を重ね，最終的に実演を行ったワークショップを構築した．今後は，本ワークショップをさらに発展させるため，以下の 4 点に取り組む予定である．1 つ目は，まちづくりに関わる多様な立場の人々から意見やアイデアを収集し，それらを可視化することである．ワークショップを通じて得られた意見を整理・共有することで，参加者間で共通のイメージを形成し，今後のまちづくり施策へと反映させることを目指す．2 つ目は，本サービスの他地域への応用である．本システムは，特定の地域データに強く依存せず，ペーパークラフトと AR を組み合わせるという特性を持つため，地域を問わず柔軟に適用できると考えられる．そのため，棒二森屋周辺と同様に再開発を検討している地域においても，意見収集や合意形成の手法として活用できる可能性がある．3 つ目は，操作性の向上による参加のしやすさの改善である．現状では，操作を担当者が個別に説明する必要があり，参加者にとっての負担となっている．今後は，直感的に操作できる UI 設計やガイド機能の導入により，「説明書を必要としない」操作性の実現を目指す．4 つ目は，AI を用いた画像生成機能の導入である．AI による画像生成を活用することで，絵を描くことに苦手意識を持つ参加者であっても，自身のイメージを容易に表現できるようになると考えられる．これにより，より多様な参加者が気軽にワークショップへ参加できる環境の構築が期待される．

第 7 章　謝辞

本プロジェクトの遂行および本報告書の作成にあたり，多くの方々にご支援とご協力をいただきました．ここに深く感謝の意を表します．

まず，本プロジェクト学習を通じて終始ご指導・ご助言を賜りました安井先生，松原先生，高見先生に，心より御礼申し上げます．

次に，本研究に関連するインタビューにご協力いただいた岡本先生ならびに市役所職員の方々には，現場の視点に基づく貴重なお話やご意見をいただきました．

また，プレワークショップや最終成果発表会にご参加いただいた皆様には，貴重なご意見やご感想をいただきました．これらのフィードバックは，本サービスの改善や完成度向上に不可欠なものであり，今後の展望を考えるうえでも大きな指針となりました．

さらに，プロジェクトを共に進めたチームメンバーには，活発な議論や役割分担を通じて，多くの学びと刺激を得ることができました．困難な場面においても協力し合い，一つの成果物を作り上げることができたことに，深く感謝しています．

最後に，本プロジェクトに関わってくださったすべての方々に，改めて感謝申し上げます．併せて，本プロジェクトの実現を支える技術を生み出し，その基盤の維持・発展に尽力されているすべてのエンジニアの皆様に，心より敬意と感謝の意を表します．

参考文献

[1] Kansai Innovation Center. (2025). ミライスケッチキャッスル – MEGA CANVAS. https://megacanvas.jp/canvas_contents/mirai-sketch-castle/

[2] 志村秀明, 辰巳寛太, 佐藤滋 (2002). 目標空間イメージの編集によるまちづくり協議ツールの開発に関する研究：建替えデザインゲームによる景観形成手法の開発. 日本建築学会計画系論文集, 67(558), 219 – 226. https://doi.org/10.3130/aija.67.219_4

[3] LOWYA（ロウヤ）. (2024 – 2025). おく ROOM ®公式サイト . https://www.low-ya.com/features/okuroom

[4] Google Cloud. vibe コーディングとは. (参照 2026-01-05). https://cloud.google.com/discover/what-is-vibe-coding?hl=ja

[5] AWS. Kiro のご紹介 – プロトタイプからプロダクションまで, あなたと共に働く新しい Agentic IDE. (参照 2026-01-05). https://aws.amazon.com/jp/blogs/news/introducing-kiro/

[6] GitHub. Spec Kit. (参照 2026-01-05). https://github.com/github/spec-kit

付録

付録 A 最終発表ポスター