ヴェリタスⅡ

飲食店シミュレーションソフト “Helios”の開発

神奈川県立厚木高等学校 2C　β 9班

目次

背景

01

背景

飲食店業界の廃業率の高さが深刻

- 採算性の見通しの甘さ
- 人流の影響や効率の悪い配置
- デッドスペースが多い
- 事前計画の甘さ

自作シミュレーションによる分析　　各項目の利益に対する影響度の特定

目的

02

目的

飲食店シミュレーションソフト　” Helios ” を開発し、
飲食店経営における各要素の利益に対する影響度の特定、分析をする。

前提条件

03

前提条件

【定義】

・従業員稼働率 ＝ 営業時間内に店員が動いた（ 配膳対応等 ）割合

・RevPASH = 売り上げ ÷（ 利用可能席数 × 営業時間 ）
（利用可能な席 1席あたり・1時間あたりに生み出された売上額）

・閾値 ＝（ 1人の店員が対応できる最大卓数 ）

前提条件

【諸設定】

・現実の1分 ＝ シミュレーション上の 1秒
・トイレに行く確率　＝30%
・ドア、厨房、トイレ　＝ レイアウト1(p)
・営業時間 ＝ 8時間（ 480秒 ）
・閾値 ＝ 4卓

前提条件

・本実験では、業態に依存しない普遍的な検証を行う。
そのため、店舗規模を以下の通り「専有面積」のみに基づいて定義する。

業態：カフェやレストラン、食べ放題など

・小規模　　80㎡（ 8m × 10m ）

・中規模　　150㎡（ 10m × 15m ）

・大規模　　400㎡（ 20m × 20m ）

前提条件　基準正規化平均感度

・本研究では、各値が利益に与える影響を判断するために、
基準正規化平均感度 という指標を導入している。

基準正規化平均感度

全施行データが基準値からどれだけ離れているかを平均したもの

前提条件　基準正規化平均感度 ①基準値の設定

表1〈各店舗の基準設定〉

※レイアウトは 1 に固定

店舗規模	従業員数 (人)	卓数 (卓)	来客数 (人)	客単価 (円)	滞在時間 (分)	総利益 (円)
小規模	2	8	60	1000	40	38,760
中規模	4	16	120	2500	60	220,000
大規模	8	32	240	4000	90	612,000

前提条件　基準正規化平均感度 ②計算式

$$変動率 = |(各利益 - 基準利益) \div 基準利益| \times 100 \quad [\%]$$

前提条件　基準正規化平均感度 ③平均感度

全データの変動率を算出し、平均をとる

▽

各項目の持つ影響力を特定

【 注意 】
・パターン④の場合、1.5，2.0 mのそれぞれで平均を出しその平均を取った
・稼働率、RevPASH はパターン①～④それぞれで平均を取った

方法

04

方法　シミュレーションソフトについて

【Helios】

・VScodeをベースにAIエージェントと我々 9班が独自開発した、
飲食店運営のリアルタイム統合シュミレーションソフト

・最大の独自性は、3Dレイアウトの変更が「人流・配膳導線・滞在時間・
売り上げKPI（RevPASH）」に同時連携する閉ループ設計

・「数理モデル・仮想空間・データ分析」を単一エンジンで統合し、
運用改善を定量比較できる

・実店舗で施行する前に、施策ごとの利益構造の可視化を実現

方法　Heliosの構成について

【開発環境】

- Node.js 20.x/npm 10.9.0/ESM
  - Vite 5.1.4（高速開発・高速ビルド）
  - Typescript strict（型の安全性向上）

【使用言語】

- Typescript/TSX
- CSS/HTML
- Javascriptconfig(cjs,mjs)
- Shell/Batch

【技術スタック】

- React＋Typescript（UI基盤）
- Three.js＋R3F＋Drei（3Dシュミレーション基盤）
- Zustand（リアルタイム状態同期）
- Recharts（KPI可視化）

方法　開発手法について

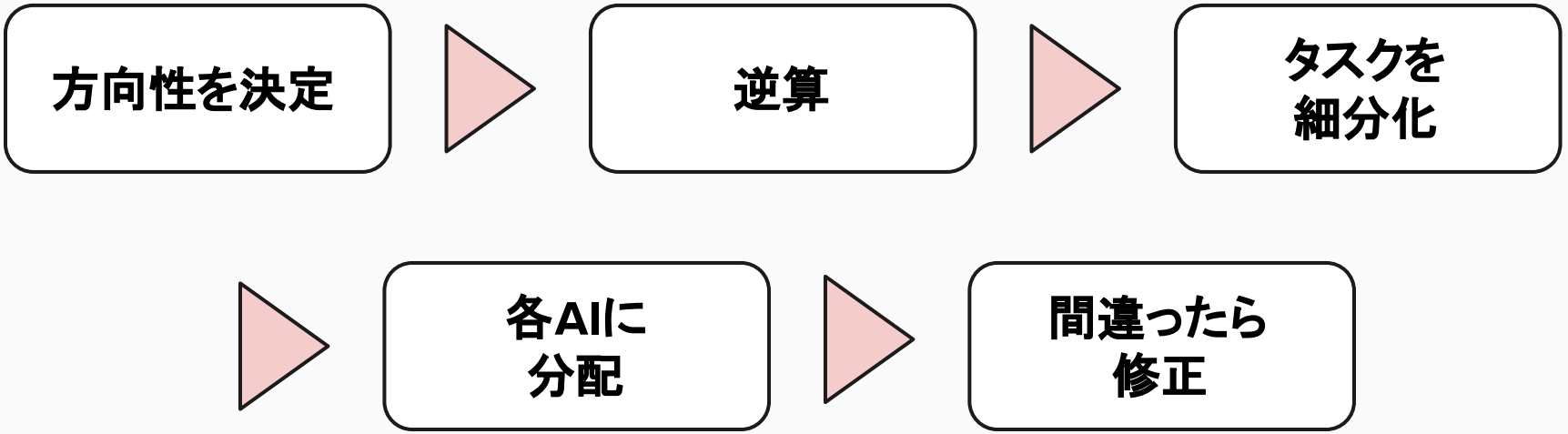
【使用したAIエージェント】

- ChatGPT5.2Thinking/GPT-5.2-Codex
  →コーディング、デバッグ等
- Gemini3Pro
  →Heliosと公的データの一致率の検証、研究手法の妥当性の検証等
- Claude Opus4.5

方法　開発手法について

【開発手法】

開発環境、使用言語、構成案




Google Gemini
判断ミスの原因と対策
ヴェルセル
Helios
helios-ifhm.vercel.app
コントロールパネル
RUNNING
店舗の大きさ
20.0×20.0m
座席設定
80席
値段設定
60.0人/単位 → 1,500円
時間設定
30.0s
従業員設定
負荷 3.00 → 係数 1.00
パラメーター設定
k 0.05 / R 0.10
店内配置設定
右 / 左
モニタリング
売上 21,000円
待機中: 0
店内: 23
着席中: 12
退店中: 4
退店済: 14
回転率: 0.17
基準来客数: 60
実効来客数: 60.00
稼働卓数: 12
店員数: 4
負荷(卓/人): 3.00
閾値: 4.00
提供時間係数: 1.00
平均滞在時間: 30.0秒
時間制: OFF
時間制上限: -
実効客単価: 1,500円
RevPASH: 2.051 円/席/秒
従業員稼働率: 74%
総売上 21,000 円
売上 = 実効客単価 × 累積着席人数

方法　シミュレーションソフトについて

・店舗の「売上（日商・月商）」の妥当性　　【 一致率：80% 】

店舗規模	Heliosの平均「純利益(日商)」	Heliosの平均月商(日商×30日)	日本政策金融公庫の統計(月商)	一致率
小規模	約43478円	約130.4万円	150万～200万円	75%
中規模	約196444円	約589.3万円	600万～800万円	84%

表2　売上の概要

■参照した出典データ　日本政策金融公庫（ JFC ）:『飲食店経営実態調査』

方法　シミュレーションソフトについて

労働生産性（ 店員1人が稼ぐ力 ）の妥当性　【 一致率：94% 】

店舗規模	店員数	Heliosの平均「純利益(日商)」	一人あたりの生産性(算出値)	経済省統計の目安	一致率
小規模	2名	約43478円	約2,17万円	約2.5万円	87%
中規模	4名	約196444円	約4,91万円	約4,5～5万円	97%
大規模	8名	約548272円	約6,85万円	約6.5～7.5万円	98%

表3　労働生産性の概要

■ 参照した出典データ　経済産業省:『経済センサス-活動調査』

方法　シミュレーションソフトについて

テーブル配置と収益改善の妥当性　【一致率：78％】

改善フェーズ	Heliosの利益 (小・中・大平均)	利益の増加率 (算出値)	コーネル大学の理論値	一致率
ステップ① (基準)	約262,732円	基準(±0%)	基準値	・
ステップ③ (時間制限)	約302,544円	＋約15.1%	10%～15% 改善	79%
ステップ④ (レイアウト)	約340,111円	＋約23%	12%～25% 改善	76%

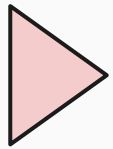
表4　テーブル配置と収益の概要

■ 参照したデータ『RevPASH（ 収益最大化 ）理論』世界　コーネル大学（ 米 ）

方法　シミュレーションソフトについて

以上の権威ある各データとの一致率…

平均 84% を記録



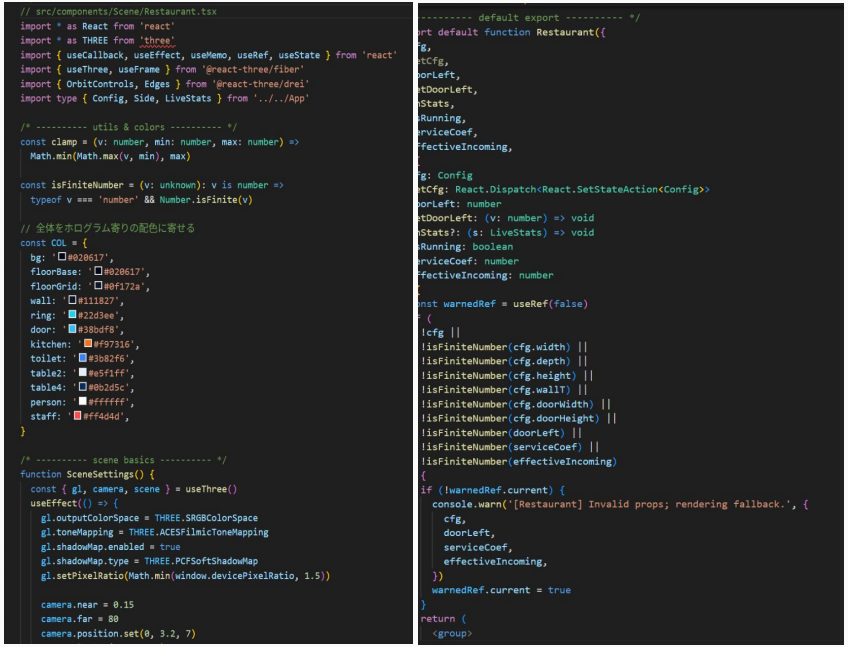
十分高く、信頼性があると判断

方法　Helios について

〈3Dレイアウト〉

- 人の動き
- 座席配置
- トイレ、厨房、ドアの位置

に関するプログラムコード



```tsx
// src/components/Scene/Restaurant.tsx
import * as React from 'react'
import * as THREE from 'three'
import { useCallback, useEffect, useMemo, useRef, useState } from 'react'
import { useThree, useFrame } from '@react-three/fiber'
import { OrbitControls, Edges } from '@react-three/drei'
import type { Config, Side, LiveStats } from '../../App'

/* --------- utils & colors --------- */
const clamp = (v: number, min: number, max: number) =>
  Math.min(Math.max(v, min), max)

const isFiniteNumber = (v: unknown): v is number =>
  typeof v === 'number' && Number.isFinite(v)

// 全体をホログラム寄りの配色に寄せる
const COL = {
  bg: '#020617',
  floorBase: '#020617',
  floorGrid: '#0f172a',
  wall: '#111827',
  ring: '#22d3ee',
  door: '#38bdf8',
  kitchen: '#f97316',
  toilet: '#3b82f6',
  table2: '#e5f1ff',
  table4: '#0b2d5c',
  person: '#ffffff',
  staff: '#f4d4d',
}

/* --------- scene basics --------- */
function SceneSettings() {
  const { gl, camera, scene } = useThree()
  useEffect(() => {
    gl.outputColorSpace = THREE.SRGBColorSpace
    gl.toneMapping = THREE.ACESFilmicToneMapping
    gl.shadowMap.enabled = true
    gl.shadowMap.type = THREE.PCFSoftShadowMap
    gl.setPixelRatio(Math.min(window.devicePixelRatio, 1.5))

    camera.near = 0.15
    camera.far = 80
    camera.position.set(0, 3.2, 7)
```

```tsx
--------- default export --------- */
rt default function Restaurant({
g,
tCfg,
orLeft,
tDoorLeft,
Stats,
Running,
rviceCoef,
ffectiveIncoming,

g: Config
tCfg: React.Dispatch<React.SetStateAction<Config>>
orLeft: number
tDoorLeft: (v: number) => void
Stats?: (s: LiveStats) => void
Running: boolean
rviceCoef: number
ffectiveIncoming: number

nst warnedRef = useRef(false)
 (
!cfg ||
!isFiniteNumber(cfg.width) ||
!isFiniteNumber(cfg.depth) ||
!isFiniteNumber(cfg.height) ||
!isFiniteNumber(cfg.wallT) ||
!isFiniteNumber(cfg.doorWidth) ||
!isFiniteNumber(cfg.doorHeight) ||
!isFiniteNumber(doorLeft) ||
!isFiniteNumber(serviceCoef) ||
!isFiniteNumber(effectiveIncoming)
{
if (!warnedRef.current) {
  console.warn('[Restaurant] Invalid props; rendering fallback.', {
    cfg,
    doorLeft,
    serviceCoef,
    effectiveIncoming,
  })
  warnedRef.current = true
}
return (
  <group>
```

図1 プログラムコード（一部抜粋）

方法　Helios について

〈コントロールパネル〉

各数値を実際に入力するパネル

に関するプログラムコード

図2 コントロールパネル

```
export default function ControlPanel({
                    step={0.1}
                    min={0}
                    value={Number.isFinite(doorLeft) ? Number(doorLeft.toFixed(2)) : 0}
                    onCommit={n => setDoorLeft(n)}
                    className="input"
                  />
                  <div className="sub">3D画面の青いドアをドラッグでも移動可能</div>
                </label>
              </div>
            )}
          </div>

          <div className="acc">
            <button className="acc-head" onClick={() => toggle('seats')} aria-expanded={open.seats}>
              <span className="acc-label">座席設定</span>
              <span className="acc-summary">{summarySeats}</span>
            </button>
            {open.seats && (
              <div className="acc-body">
                <div className="row2">
                  <label className="field">
                    <div className="label">4人席の数</div>
                    <DraftNumberInput
                      min={0}
                      step={1}
                      value={cfg.table4Count}
                      onCommit={n => setNum('table4Count', n)}
                      className="input"
                    />
                  </label>
                  <label className="field">
                    <div className="label">2人席の数</div>
                    <DraftNumberInput
                      min={0}
                      step={1}
                      value={cfg.table2Count}
                      onCommit={n => setNum('table2Count', n)}
                      className="input"
                    />
                  </label>
                </div>
              </div>
            )}
```

図3 プログラムコード（一部抜粋）

方法　Helios について

〈Helios全体〉

・3Dレイアウト
・コントロールパネル
　などを仲介する機構

に関するプログラムコード

```
const normalizeConfig = (
  const normalized = {
    repulsion: {
    },
  }
  if (warn && warnKeys.length) {
    warn(Array.from(new Set(warnKeys)))
  }
  return normalized
}

export default function App() {
  const [cfg, setCfg] = useState<Config>(() => normalizeConfig(DEFAULT_CONFIG))
  const warnOnceRef = useRef(new Set<string>())
  const warnKeysOnce = useCallback((keys: string[]) => {
    const fresh = keys.filter(key => !warnOnceRef.current.has(key))
    if (!fresh.length) return
    fresh.forEach(key => warnOnceRef.current.add(key))
    console.warn('[config fallback]', fresh)
  }, [])
  const warnValueOnce = useCallback((key: string, value: unknown) => {
    if (warnOnceRef.current.has(key)) return
    warnOnceRef.current.add(key)
    console.warn(`[fallback] ${key}`, value)
  }, [])

  const safeCfg = useMemo(() => normalizeConfig(cfg, warnKeysOnce), [cfg, warnKeysOnce])

  useEffect(() => {
    setCfg(p => {
      if (p.currentPrice === p.avgSpend) return p
      if (p.currentPrice == null) return p
      return { ...p, avgSpend: p.currentPrice }
    })
  }, [cfg.currentPrice])

  // Door left edge position.
  const initialDoorLeft = useMemo(
    () => (safeCfg.width - safeCfg.doorWidth) / 2,
    [],
  )
  const [doorLeft, setDoorLeft] = useState(initialDoorLeft)

  useEffect(() => {
```

```
export default function App() {
            <Restaurant
              cfg={safeSceneCfg}
              setCfg={setCfg}
              doorLeft={safeDoorLeft}
              setDoorLeft={setDoorLeft}
              onStats={setStats}
              isRunning={isRunning} // required for simulation timing
              serviceCoef={serviceCoef}
              effectiveIncoming={effectiveIncoming}
            />
          </Suspense>
        </Canvas>

        {/* Control panel */}
        <div
          style={{
            position: 'fixed',
            left: 24,
            top: 24,
            zIndex: 10,
            border: '1px solid rgba(0,255,255,0.3)',
            background: 'rgba(5,10,25,0.85)',
            boxShadow:
              '0 0 25px rgba(0,255,255,0.18), 0 0 80px rgba(0,180,255,0.22)',
            backdropFilter: 'blur(20px)',
            borderRadius: '22px',
            padding: '16px 16px 20px 16px',
            overflowX: 'hidden',
          }}
        >
          <ControlPanel
            cfg={safeCfg}
            setCfg={setCfg}
            doorLeft={safeDoorLeft}
            setDoorLeft={(v: number) => {
              const m = 0.1
              setDoorLeft(
                clamp(v, m, Math.max(m, safeCfg.width - safeCfg.doorWidth - m)),
              )
            }}
            stats={{ ...stats, revPASHPerSec }}
            totalSales={totalSales}
            elapsedSeconds={elapsedSeconds}
```

図4 プログラムコード（一部抜粋）

実験①　ステップ１

1, 座席数(オペレーション負荷)実験

卓数を変更したことによる利益の変化

操作:
- 卓数を基準値から上下に1卓ずつ変えて実行。
- ±１卓　±２卓　±３卓に変動
- その他の項目を固定し、対照実験を行う
- 今回は四人座席のみ活用する。

図2 コントロールパネル

実験①　ステップ１　（例）

図5 8卓の場合

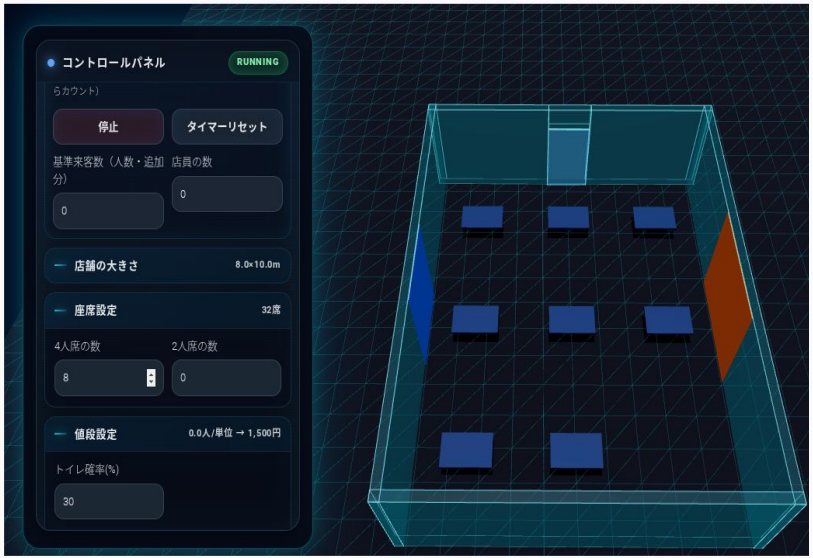
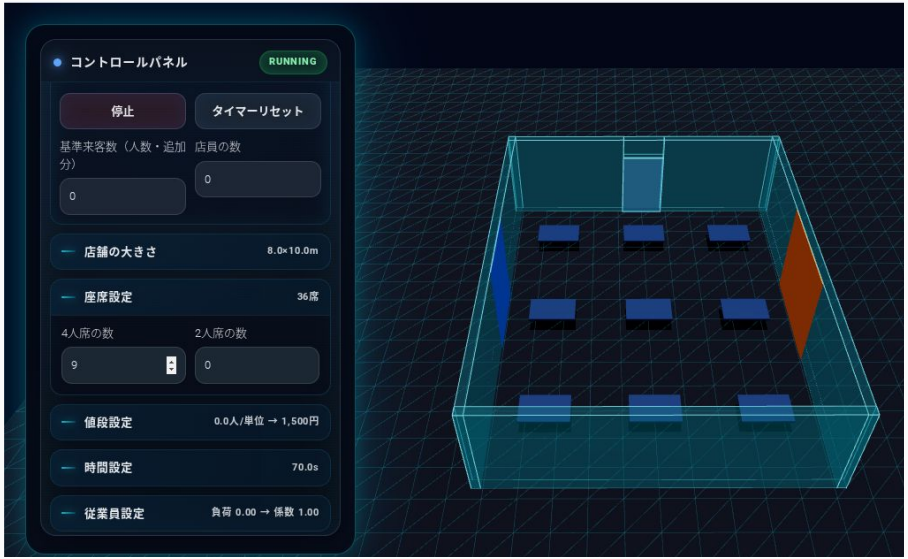


図6 9卓の場合



※座席配置を変更した時の影響はほとんどない

実験①　ステップ２

2，価格弾力性実験

価格の変動に対しての来客数の増減

操作：
・基準価格を基本パターンに固定する
・現在価格の項目を基準価格から 10%ずつ変える
・± 10%、± 20%、± 30%に変動
・その他の項目を固定し、対照実験を行う

図2 コントロールパネル

実験①　ステップ３

3，制限時間実験

制限時間の変動で、客単価の変化の影響

操作：
・基準滞在時間を基本パターンに固定する
・時間制上限の項目を基準価格から 10%ずつ変える
・± 10%、± 20%、± 30%に変動
・その他の項目を固定し、対照実験を行う

図2 コントロールパネル

実験②　人流（ レイアウト ）の比較実験

レイアウトと座席間隔を変更して、
人流を変化によって利益にどう差が出てくるかを調べる

異なるレイアウト　11通り
＋
座席間隔　1.5m

異なるレイアウト　11通り
＋
座席間隔　2m

歩く距離 や
客同士の干渉 が変わり、
それが利益を左右する

実験②　人流(レイアウト)の比較実験

レイアウト11通りの概要　　青:トイレ　橙:厨房

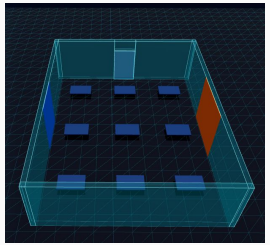


図7 1【左　右】

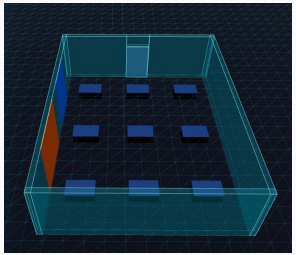


図8 2【左　左】

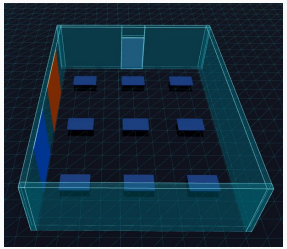


図9 3【左　左】

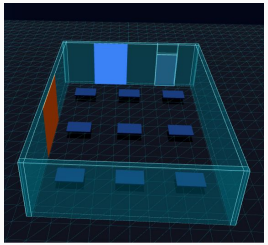


図10 4【奥　左】

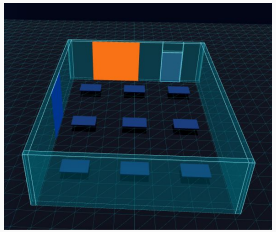


図11 4【左　奥】

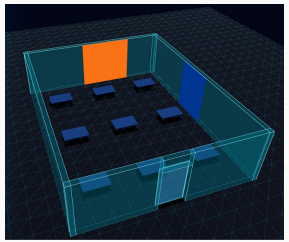


図12 5【左　奥】

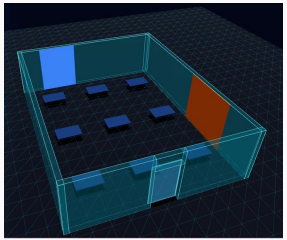


図13 6【手前　左】

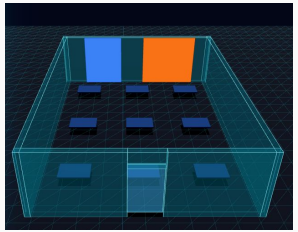


図14 7【手前　手前】

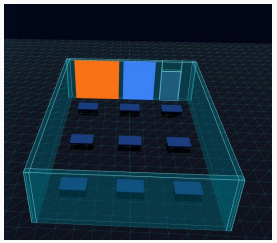


図15 8【奥　奥】

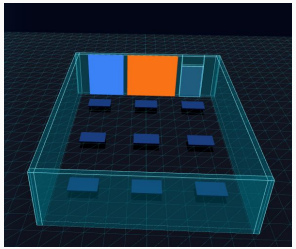


図16 9【奥　奥】

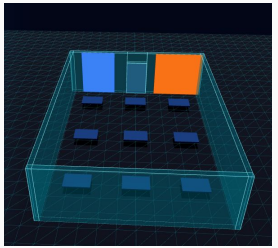


図17 10【奥　奥】

実験①②

それぞれのステップにおいて、モニタリング項目を記録する

図18　モニタリング

主にデータとして利用する項目

- 従業員稼働率
- RevPASH
- 総利益
- 各変数パラメータ

結果

05

結果　全体のグラフ

図19 利益感度と稼働率

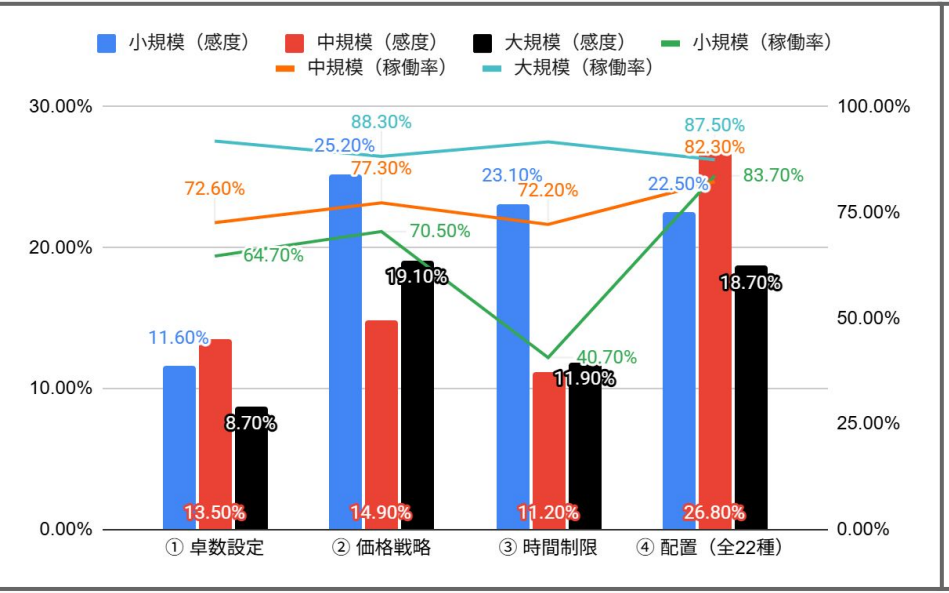
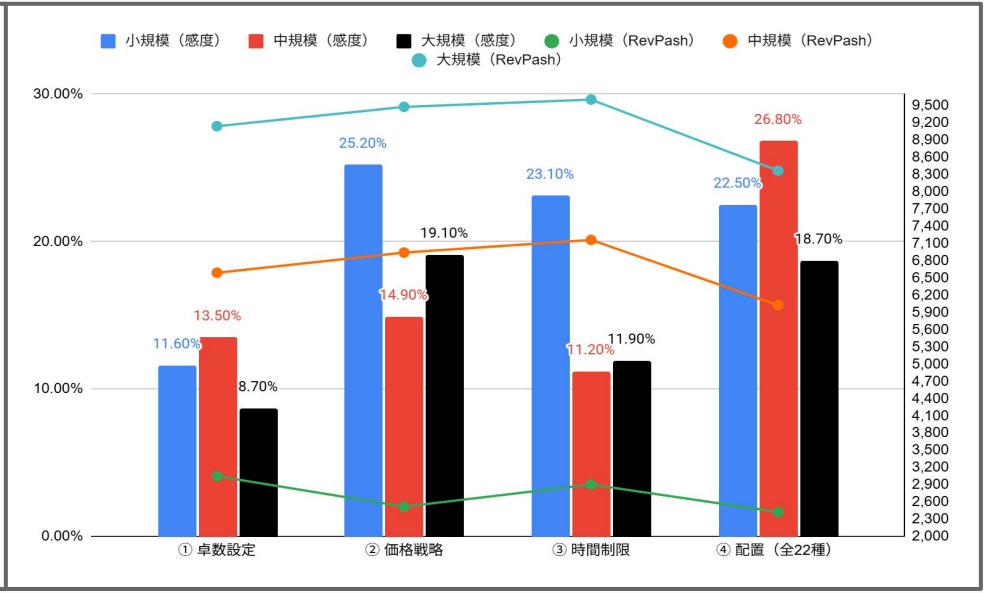


図20 利益感度とRevPASH



考察

06

考察　小規模店

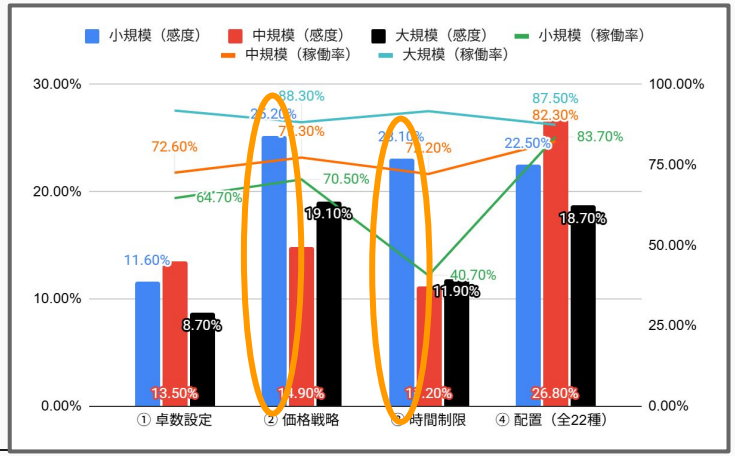
影響の大きい項目

・客単価
・滞在時間

▷ 大・中規模店と比べ **面積が小さいため**

▽

・客単価を上げること
・滞在時間を短くし回転率を上げること



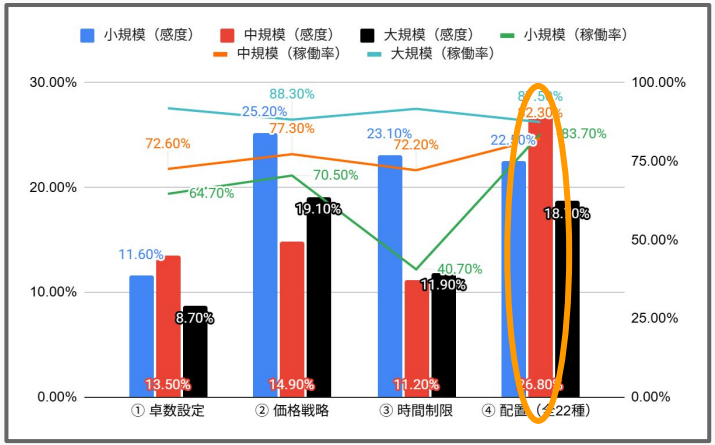
考察　中規模店

影響の大きい項目

・配置（レイアウト）

大規模店と比べ **面積が小さい**
小規模店と比べ **卓数が多い**

空きスペースを利用した
より効果的な座席配置の検討

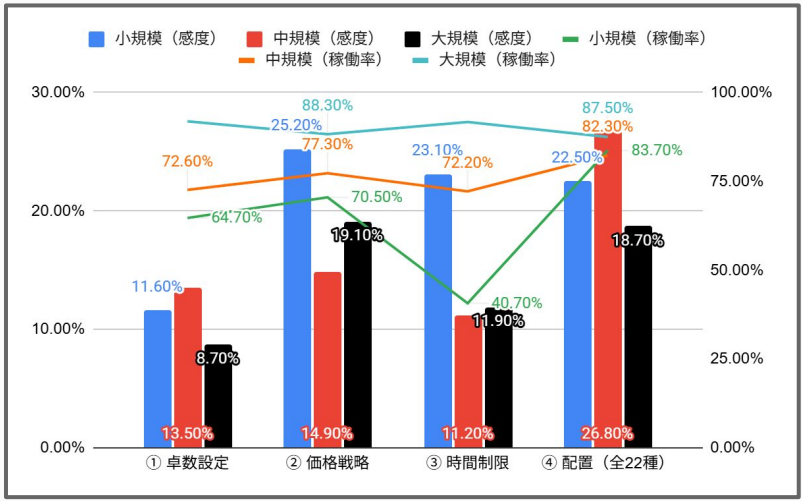


考察　大規模店

各値を変更したときの 利益の差は他店舗より 小さい

▽

新しく施策をするよりも、
ミスをしないことが重要

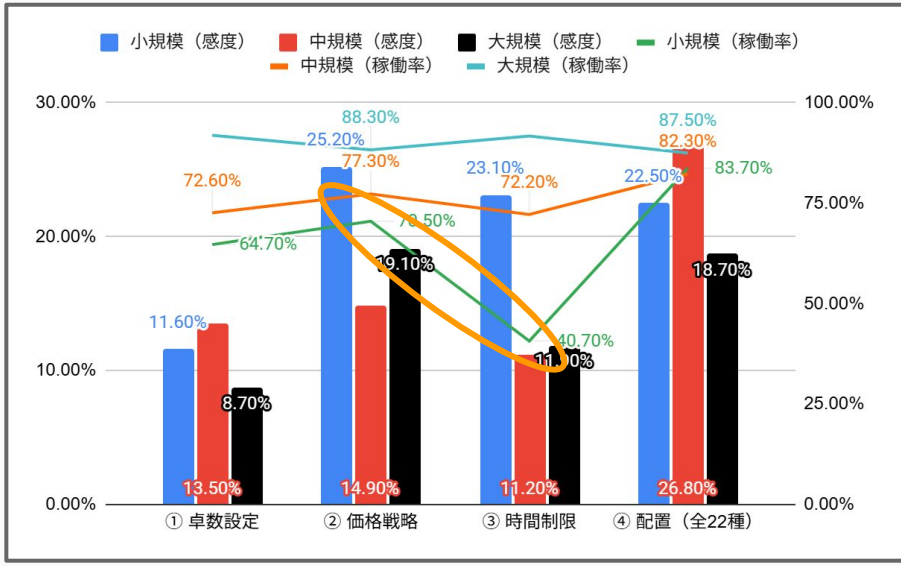


考察　総評①

小規模店

平均稼働率の下がっている箇所がある

滞在時間の短縮 → 追加注文減少
→ 店員が卓に戻る回数が減る

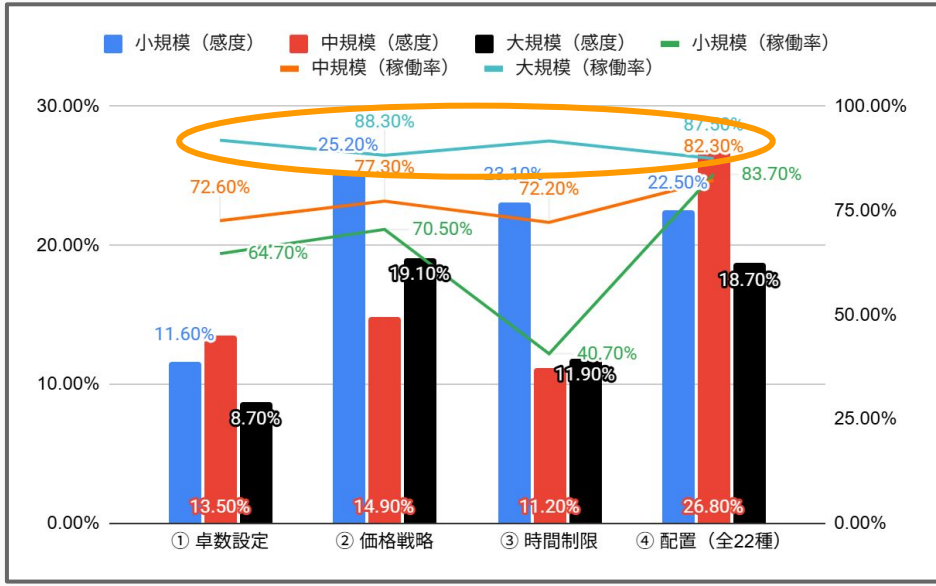


考察　総評①

大規模店

平均稼働率に大きな差は見られない

小規模に比べ面積が大きい
→ 配膳一回の負担が大きい



考察　総評 ②

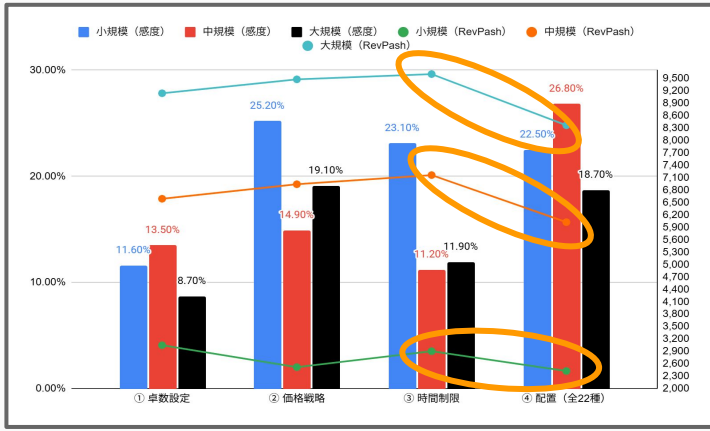
どの店舗もステップ④における　RevPASH の値が
右肩下がりになっている

▽

ステップ④だけは試行回数が22回と多く
効率の悪い配置も含まれている

↓

平均値を下げる原因



結論

07

結論

・一般的な利益の向上
単価を上げることや人を減らし、人件費を下げること

・今回の研究
飲食店という一つのくくりの中でも小・中・大規模店において
利益感度や従業員稼働率、ReVPASHで異なる結果が得られた。

各店が持つ特性（大規模・小規模なのか等）にアプローチするような戦略　を立案することが効率的かつ合理的に飲食店というビジネスを成功に導くのではないかと考える。

今後の展望

08

今後の展望

飲食店との提携

実際の飲食店提携のフィードバックを元に、
現実との互換性を向上させる

↓

飲食店の起業希望者が起業前に
自分の店の間取りで何度もシミュレーション可能

↓

意思決定をサポートするようなプロダクトへ

今後の展望

異なる人流シミュレーションへの応用

飲食店営業に留まるだけでなく、
人の動きをシミュレートできることを身近なものに活かす

（例）
・ショッピングモールや学校での緊急時の人流の可視化
・医療におけるナースコール対応に対応しやすい配置
・オフィスや教育現場の動き

今後の展望

Helios自体のロジックの改善・強化

図21　Helios2

Heliosのver2を開発中。

Helios2では、
今回のようなに人を簡易に再現するのではなく、
人型モデルへ改良し、着席する動作や実際に配膳する動作などが追加される。
また、店舗 UIは今回のような簡素な四角形ではなく、階段・カウンター・ドリンクバー
など実際の店舗を写真撮影すると反映されるというシステムを開発中である。

参考文献

社会課題の解決につながるDXへ
株式会社日立製作所　2025/10/6閲覧
https://www.hitachihyoron.com/jp/archive/2020s/2021/03/activities/index.html

人流解析技術を用いたビル内移動の最適化シミュレーション
株式会社日立製作所　2025/10/6閲覧
https://www.hitachihyoron.com/jp/archive/2010s/2018/02/02a03/index.html

汎用的な人流シミュレーションシステム
国土交通省　2024　2025/10/6閲覧
https://www.mlit.go.jp/plateau/use-case/uc24-07/

参考文献

コーネル大学参考データ

1. テーブル配置（テーブルミックス）の最適化に関する研究

出典: Cornell Study Finds Restaurateurs Are Leaving Money on the Table (Hospitality Net)

2. RevPASH（時間あたりの収益）の基本理論

出典: Implementing Revenue Management in Your Restaurants (Cornell eCommons)

3. 具体的な改善事例（Chevys Arrowheadの事例）

出典: Restaurant Revenue Management: Implementation at Chevys Arrowhead

4.経済産業省

経済センサス-活動調査（経済産業省サイト ）

5.日本政策金融公庫

飲食店経営実態調査結果（令和3年度）