

ゲーム開発とAI

株式会社リンクトブレイン
技術戦略顧問
増渕大輔



増渕大輔

株式会社リンクトブレイン 技術戦略顧問

オンラインゲーム技術者およびDXコンサルタント

デジタルハリウッド大学院ジーズアカデミー・メンター



- 講演実績

- CEDEC (Computer Entertainment Developers Conference)
- IDGAJ (International Game Developers Association Japan)
- Microsoft de:code
- Xbox, Xfest
- その他、各種勉強会

リンクトブレインの会社概要



企業名

株式会社リンクトブレイン
英 : Linked Brain Inc.

設立

2011年10月

所在地

本社 〒102-0083 東京都千代田区麹町3-7-4 秩父屋ビル2F
福岡クリエイティブセンター 〒812-0018 福岡市博多区住吉2-2-1
井門博多ビルイースト4F

メンバー

155名

WEB

<http://linkedbrain.jp>

取得資格

有料職業紹介事業【13-ユ-305862】

一般労働者派遣事業【派13-305515】

プライバシーマーク認定【17002715(01)】



グループ会社

株式会社ロジックボックスピクチャーズ（東京）
株式会社uzufactory（福岡）



OUR MISSION

クリエイターが活躍できる
感動にあふれた世界を共創する



ゲーム運営・ 開発受託事業

企画開発、2D、3Dデザインと
専門分野から一括受託まで、ゲ
ーム制作をお受けしています。



ゲーム業界特化型 HR事業

ゲーム業界での転職をご検討中
の方のサポートを行っています。



ゲームアプリのイ ンポート・エクス ポート事業

海外タイトルのカルチャライズ
などお受けしています。



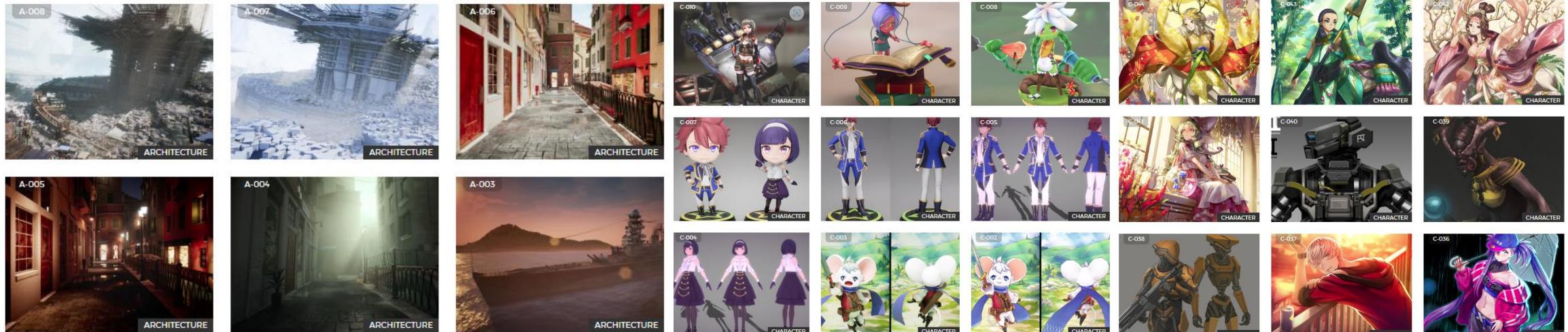
エンタープライズ xR事業

ゲーム開発経験を活かしたコン
テンツ企画開発をお受けしてい
ます。

ゲーム業界特化型 HR事業

Linked Brain

正社員・契約社員の転職支援、派遣、業務委託案件、様々なニーズに対応



ゲーム業界特化
エージェント
G エージェント
G-JOB



メタバース開発

専用のアプリ不要でWEB上で体験することのできる3Dコンテンツの開発、ならびにWEB開発に最適な3Dモデルを制作いたします。
バーチャル展示会やバーチャルオフィスなどのメタバースコンテンツの制作も、企画や仕様設計の段階からサポートいたします。

WebGL開発（Three.js / Babylon.js）/ 3DCG制作（GLB/GLTF形式）/ 企画 / ディレクション



VR・AR・MRコンテンツ開発



AR、VR、MRを使いドローンシミュレーター、内装シミュレーション、危険マニュアルなど、臨場感や没入感が感じられる高品質なユーザー体験を実現するコンテンツを制作します。企画・設計から開発までサポートすることでお客様のニーズとユーザーのニーズをマッチさせる最善のコンテンツをご提案いたします。

各種開発 / 企画 / ディレクション



CG制作

キャラクターモデルの制作からリアルな建物再現まで様々なCG制作が得意なスタッフが在籍しています。複雑なデザインも高解像度なクオリティで表現いたします。
流行りのVtuberキャラクター制作やAIに学習させるための素材(画像/動画)などのCGの制作、ドローンを使った空撮による映像制作などの実績もございます。

3DCG制作 / 機械学習素材制作 / 企画 / ディレクション

ゲーム開発

と

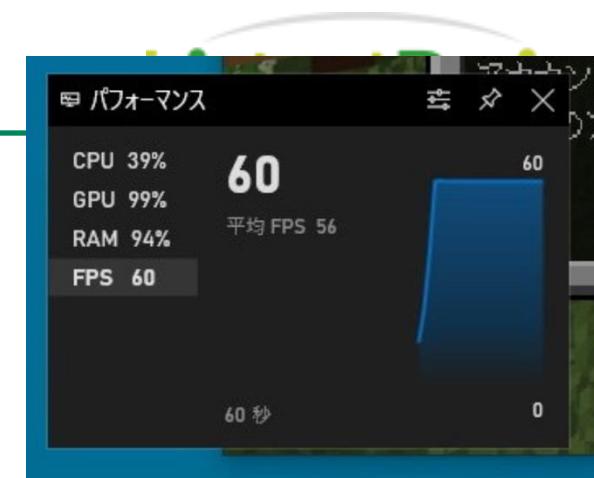
AI

- ゲーム開発とAI の概要
- ゲームにおけるAI利用の変遷と拡張
- 生成AIはゲーム開発に使えるの？
- ゲーム産業における象徴的なAI事例
- まとめと振り返り

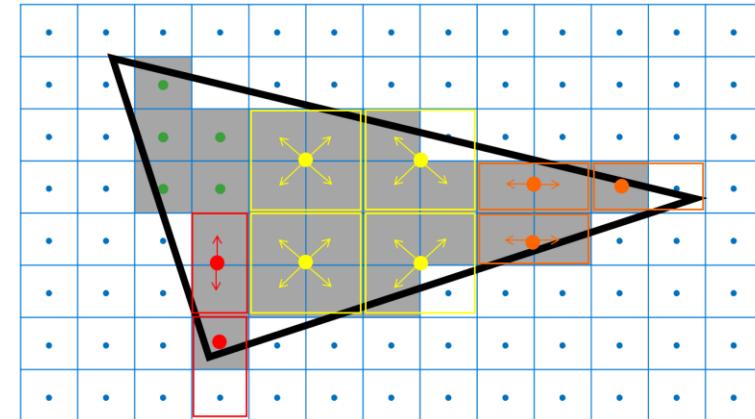
- ゲーム開発とAI の概要
- ゲームにおけるAI利用の変遷と拡張
- 生成AIはゲーム開発に使えるの？
- ゲーム産業における象徴的なAI事例
- まとめと振り返り

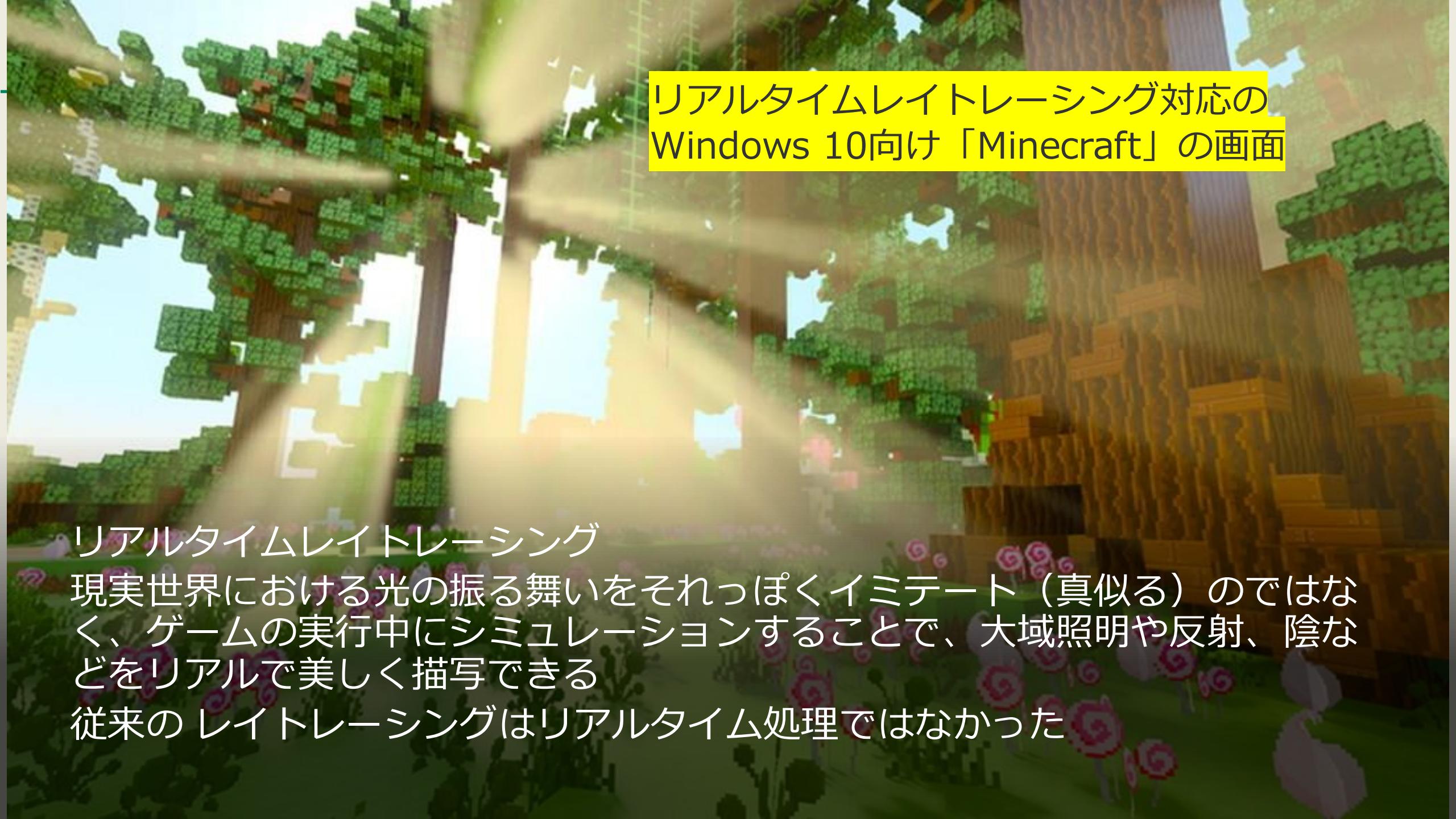
ゲームにはグラフィックスが必要

- **DirectX:** Windows 95 の発売の直前、マイクロソフトは Windows 95におけるプログラムの自由度を上げる仕組みを作り上げたのが「DirectX」
- よりハードウェア制御に近いローレベルな制御を行うことができる
- 最新の DirectX 12 Ultimate は、ゲームなどのマルティメディアアプリケーションを作成するためのAPI群
- **DirectX Raytracing 1.1 (DXR 1.1):** リアルタイムレイティングを実現するグラフィックスAPIの次期版
- **Variable Rate Shading (VRS):** シェーダーのピクセル解像度を状況に応じて変化させ、見た目に影響を与えず描画
- **Mesh Shaders:** Turing世代のNVIDIA製GPUに追加されたシェーダー
- **Sampler Feedback:** 不要な計算を省いてレンダリング負荷を下げる技術の一つ



VRS : 1x1, 1x2, 2x1, 2x2 の
解像度を使い分ける





リアルタイムレイトトレーシング対応の
Windows 10向け「Minecraft」の画面

リアルタイムレイトトレーシング

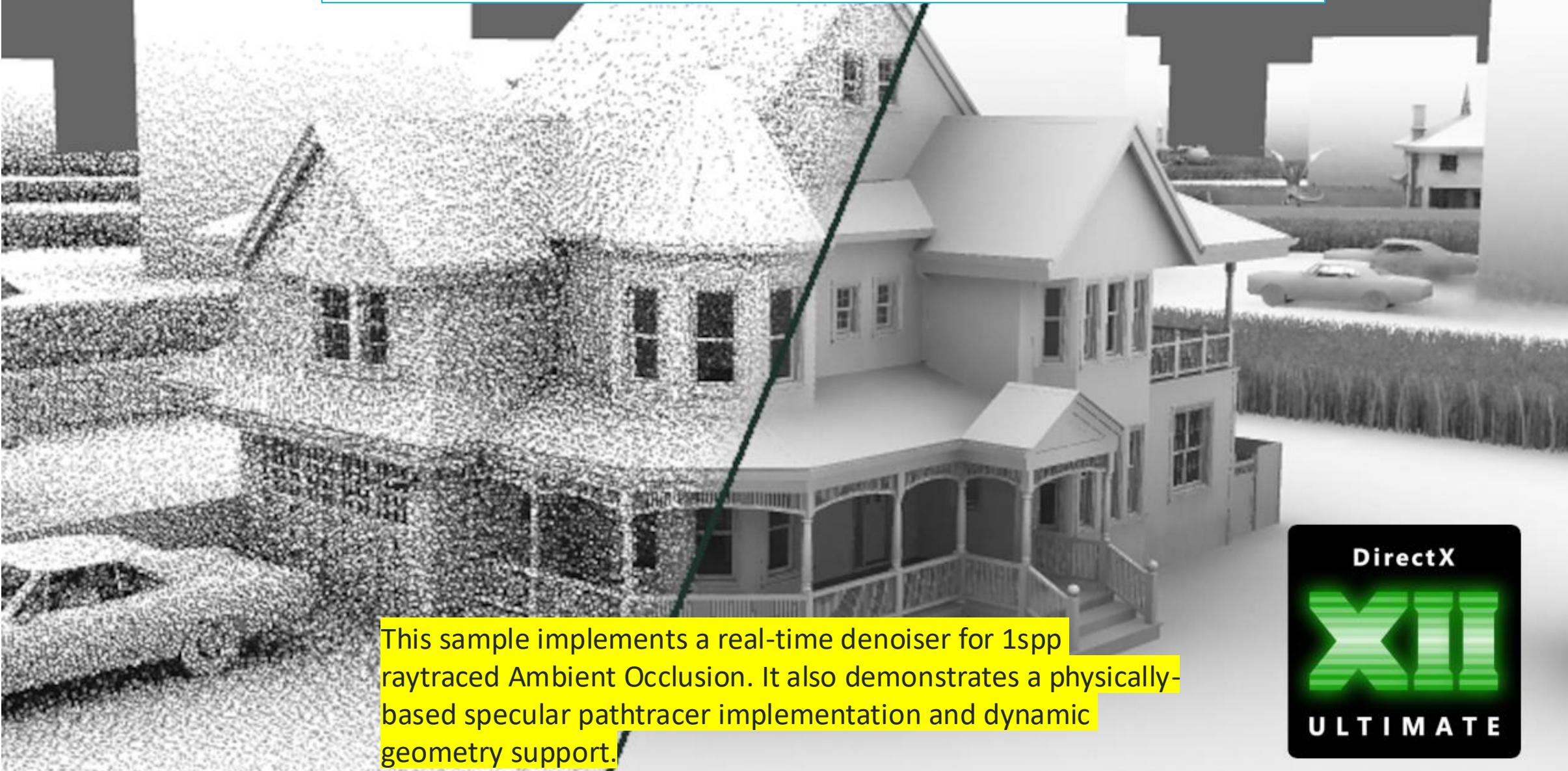
現実世界における光の振る舞いをそれっぽくイミテート（真似る）のではなく、ゲームの実行中にシミュレーションすることで、大域照明や反射、陰などを探して描写できる

従来のレイトトレーシングはリアルタイム処理ではなかった



「シヴィライゼーション5」の画面
半分だけにVRS（Variable Rate Shading：可変レートシェーディング）を適用。
VRSはハードウェアレンダリングが14%高速化しているが、見た目の品質に
違いはない

Real-Time Denoised Ambient Occlusion



ゲームの3D表現には大量の行列計算が必要

- ・ 三角形の位置、角度、色、テクスチャの計算
- ・ 1秒間に数百万回の並列計算が必要
- ・ CPUでは処理が間に合わない（1フレームに1-2秒）

GPUの特徴を確立

- ・ 単純な計算を大量に並列処理
- ・ 高速なメモリアクセス
- ・ プログラム可能な演算機能

GPUが機械学習を加速

- ニューラルネットワークは単純な行列計算
 - 画像認識は数値の行列処理
 - 学習には大量の行列乗算が必要
- GPUの特徴が完全にマッチ
 - 並列処理による高速化（数ヶ月→数時間）
 - 大容量メモリによる効率的な処理
 - プログラム可能な柔軟性

ゲーム産業がGPUを発展させ、そのGPUが機械学習を加速させる、相乗効果が生まれました。

ビデオゲーム技術がニューラルネットワークを可能にする仕組み

<https://techcrunch.com/2017/10/27/how-video-game-tech-makes-neural-networks-possible/>

- GPU がニューラルネットワークのトレーニングに優れている理由
 - GPU が行列乗算に優れているため
 - GPU は 1 つの数値テーブル (たとえば、画像のある部分のピクセルの値) を取得し、それを別のテーブルで乗算することができます (値は別の部分にあります)
 - ニューラルネットワークは行列の乗算に大きく依存しているため、GPU を使用すると、トレーニングにかかる時間がCPUでは、数か月、数週間から数日、場合によっては数時間まで短縮されます
- 最近の GPU は大量のオンボード メモリを搭載している
 - コンピュータのメインメモリとの間でデータを往復する遅延がない
 - プログラム可能なので、手書きや音声認識など、さまざまなタスクを実行する
- 2015 年、Google と Microsoft は、毎年恒例の ImageNet コンピューター ビジョン チャレンジにおいて、画像内の物体を 人間よりも正確に識別できるディープニューラルネットワークを設計
- エヌビディアは、GPU を使用したニューラルネットワークのトレーニング速度を わずか 3 年間で 50 倍 高速化したと発表
- Google は、ニューラルネットワークでの使用に特化して設計され、処理できる新しい「Tensor プロセッシング ユニット」の開発に取り組んでいる

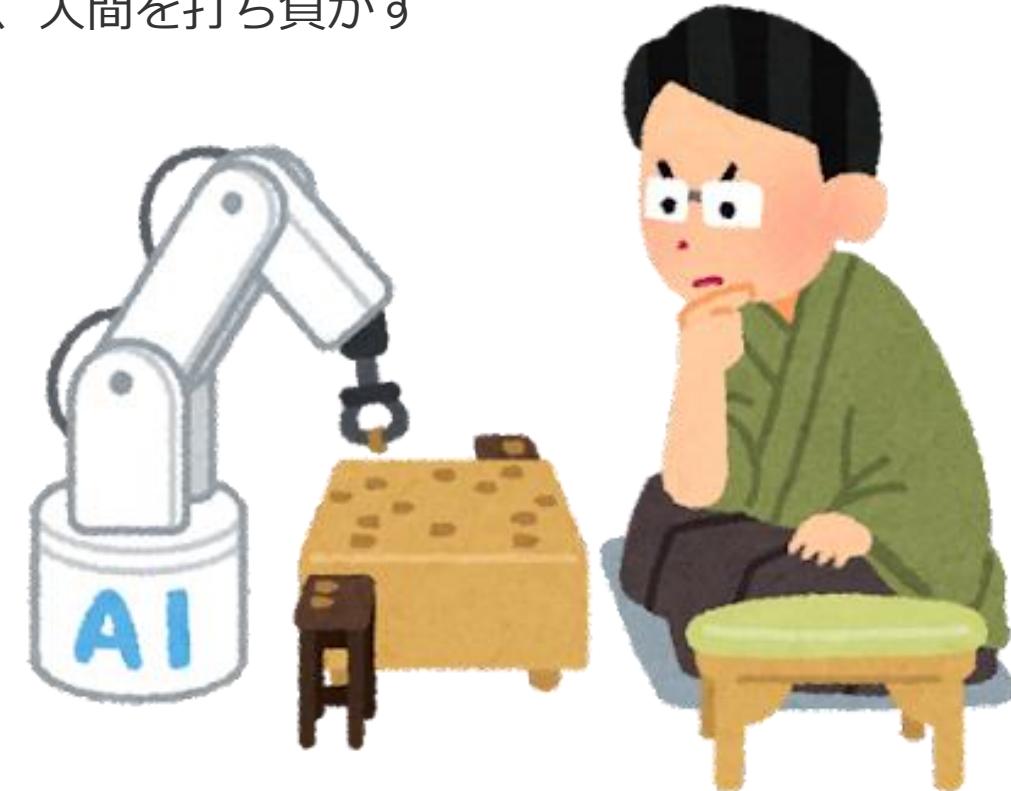
なぜゲームにAIが必要なの？？？

エネミーAIが、古くから、ゲームにおけるAIの主要な用途の1つ

- ・ コンピュータの対戦相手の振る舞い
- ・ アクションゲームにおける比較的単純なパターンから、人間を打ち負かす
チェスプログラムまで

高度なものは、生成的敵対ネットワーク(GAN)技術、
画像認識、などを用いることがある

感情を読み取り、テキストから感情を生成し、
感情を正確に描写するアルゴリズム

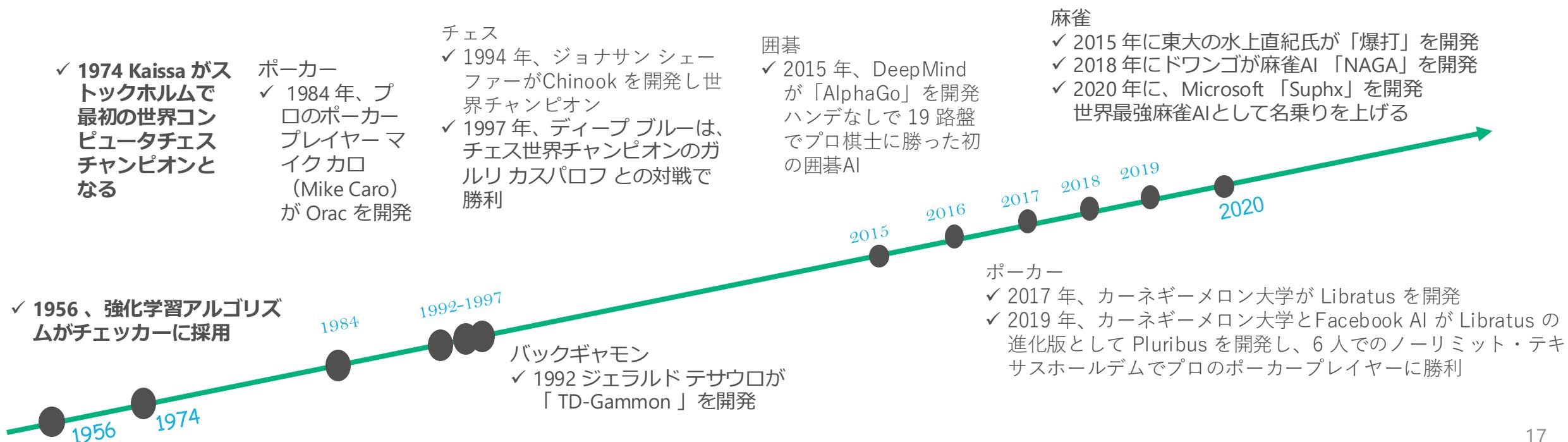


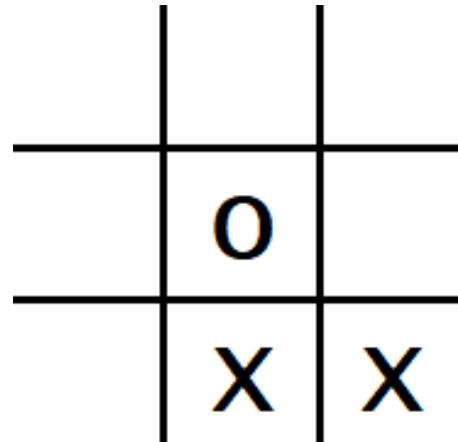
ゲーム AI の進化と歴史

コンピュータサイエンティストは、AIのアルゴリズムと問題解決能力を向上させるためにゲームAIに注目してきました（チェスや囲碁など、シンプルで明確なルールがあるゲームがAI研究対象として優れていた）

参考：Microsoft 「ゲーム AI の進化と歴史」

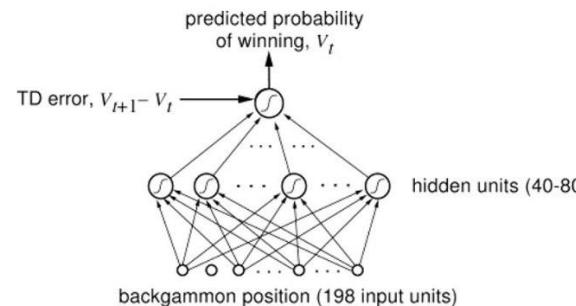
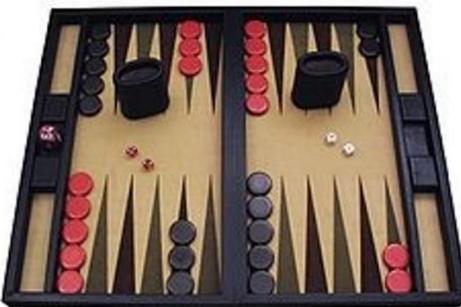
- <https://news.microsoft.com/ja-jp/2019/08/19/190819-evolution-and-history-of-game-ai/>





- 三目並べの碁盤には $9 (3 \times 3)$ の格子がある
- 各格子には X、O、空白という 3 つの状態がある
- 局面数は 3 の 9 乗である、 19863 となる
- 状態空間複雑性は約 10^4 ($19863 \approx 10^4$)

完全情報ゲーム: 状態空間複雑性とゲーム木複雑性



ゲーム	状態空間 複雑性	ゲーム木 複雑性
三目並べ	10^4	10^5
チェックター	10^{21}	10^{31}
チェス	10^{46}	10^{123}
中国象棋	10^{48}	10^{150}
五目並べ	10^{105}	10^{70}
囲碁	10^{172}	10^{360}

参考

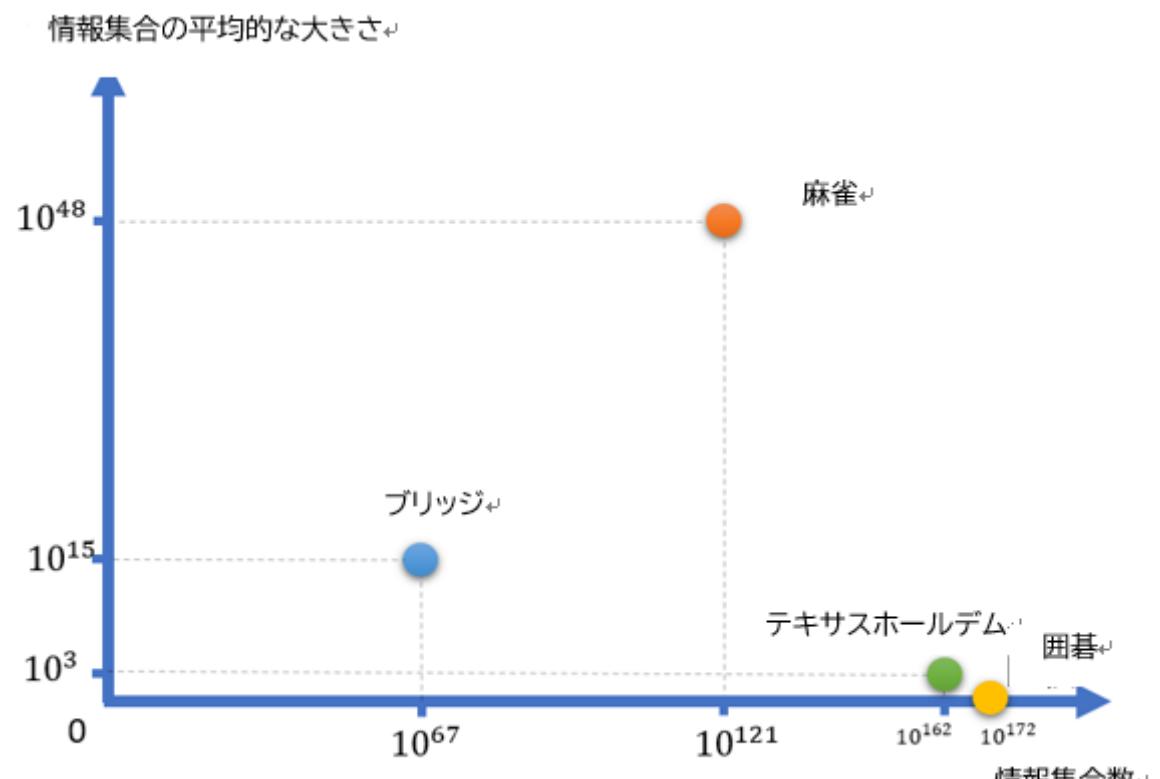
[ゲーム AI の難易度とは? - News Center Japan \(microsoft.com\)](#)

完全情報ゲーム？ 不完全情報ゲーム？

- 不完全情報ゲーム： 参加者はゲームの実際の状態を分別することができない
 - 分別できないゲームの状態の集合を情報集合といいます。
 - 合理的なゲーム戦略は、ゲームの状態ではなく、情報集合に基づいて考える
- 難易度を測りかた
 - 状態空間の大きさではなく、情報集合の数を尺度にする
 - 「完全情報ゲーム」の情報集合数は状態空間数と同じ
- 情報集合の平均的な大きさ
 - 情報集合の中で区別できないゲームの状態の平均数
 - ゲームの各局面の裏に隠されている情報の数

不完全情報ゲーム: 情報集合数と平均的な大きさ

ゲーム	情報集合数	情報集合 の平均的な大きさ
1対1テキサスホールデム (制限付き)	10^{14}	10^3
1対1テキサスホールデム (無制限)	10^{162}	10^3
ブリッジ	10^{67}	10^{15}
麻雀	10^{121}	10^{48}

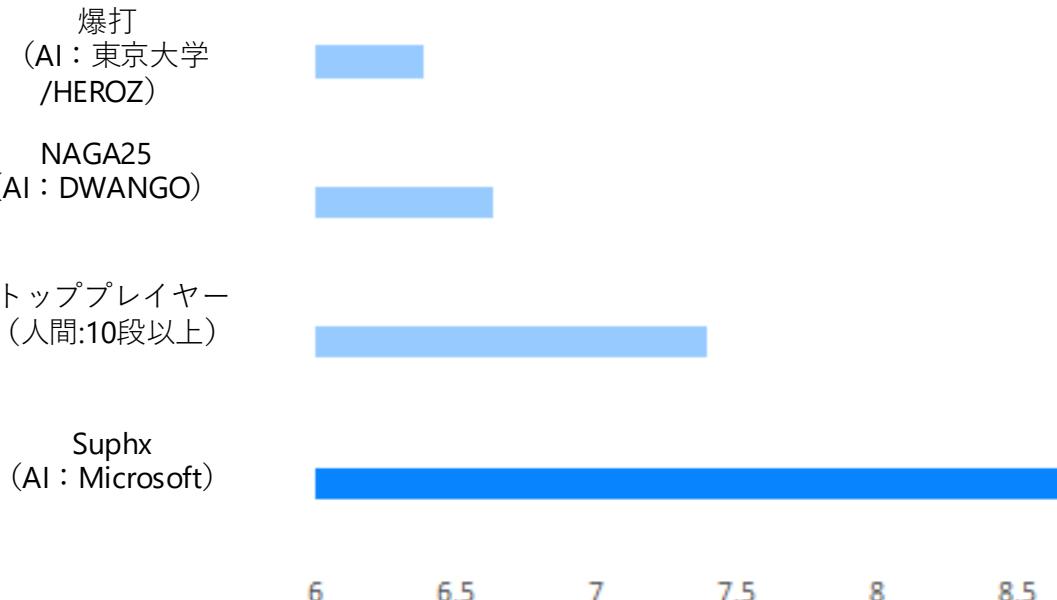


ゲームAIの難易度とは? - News Center Japan (microsoft.com)

オンライン麻雀プラットフォームの「天鳳」で10段獲得

- 「安定段位」（平均した強さ指標）において スコア8.7 を達成
- 「特上卓」に参加するトッププレイヤーとの5,000回以上の対局における平均の成績
- 人間のトッププレイヤーの平均を上回る成績

天鳳 安定段位における比較



認定

認定段位/級位：十段

⑩Suphx 殿

貴殿は天鳳において卓越した技能を遺憾なく発揮され優秀な成績をおさめられました。今後もさらなる雀力向上に精進されますようここに段位/級位を認定し栄誉を称えます。

2019年06月22日
天鳳段位認定協会

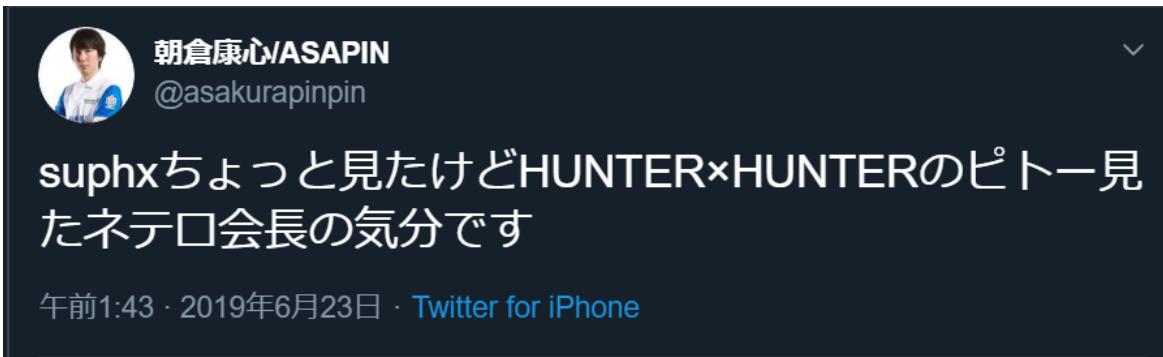
OK

有段者からの評価・評判

「ASAPIN」のプレイヤー名で知られる朝倉康心氏

<https://twitter.com/asakurapinpin>

日本の麻雀プレイヤーの中では神格的存在、世界で初めて天鳳の最高位である「天鳳位」を獲得



朝倉康心/ASAPIN
@asakurapinpin

suphxちょっと見たけどHUNTER×HUNTERのピトー見たネテ口会長の気分です

午前1:43 · 2019年6月23日 · Twitter for iPhone

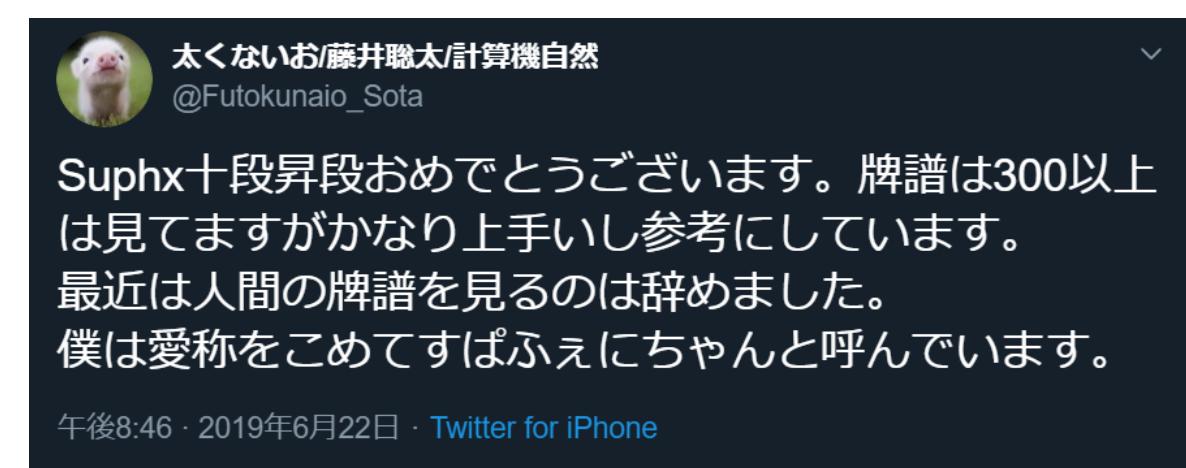
(=私より強いかもしくないと感じている)

<https://twitter.com/asakurapinpin/status/1142473124081913856?s=20>

「太くないお」氏

https://twitter.com/Futokunaio_Sota

3人打ちおよび4人打ち麻雀の両方の「天鳳位」を獲得



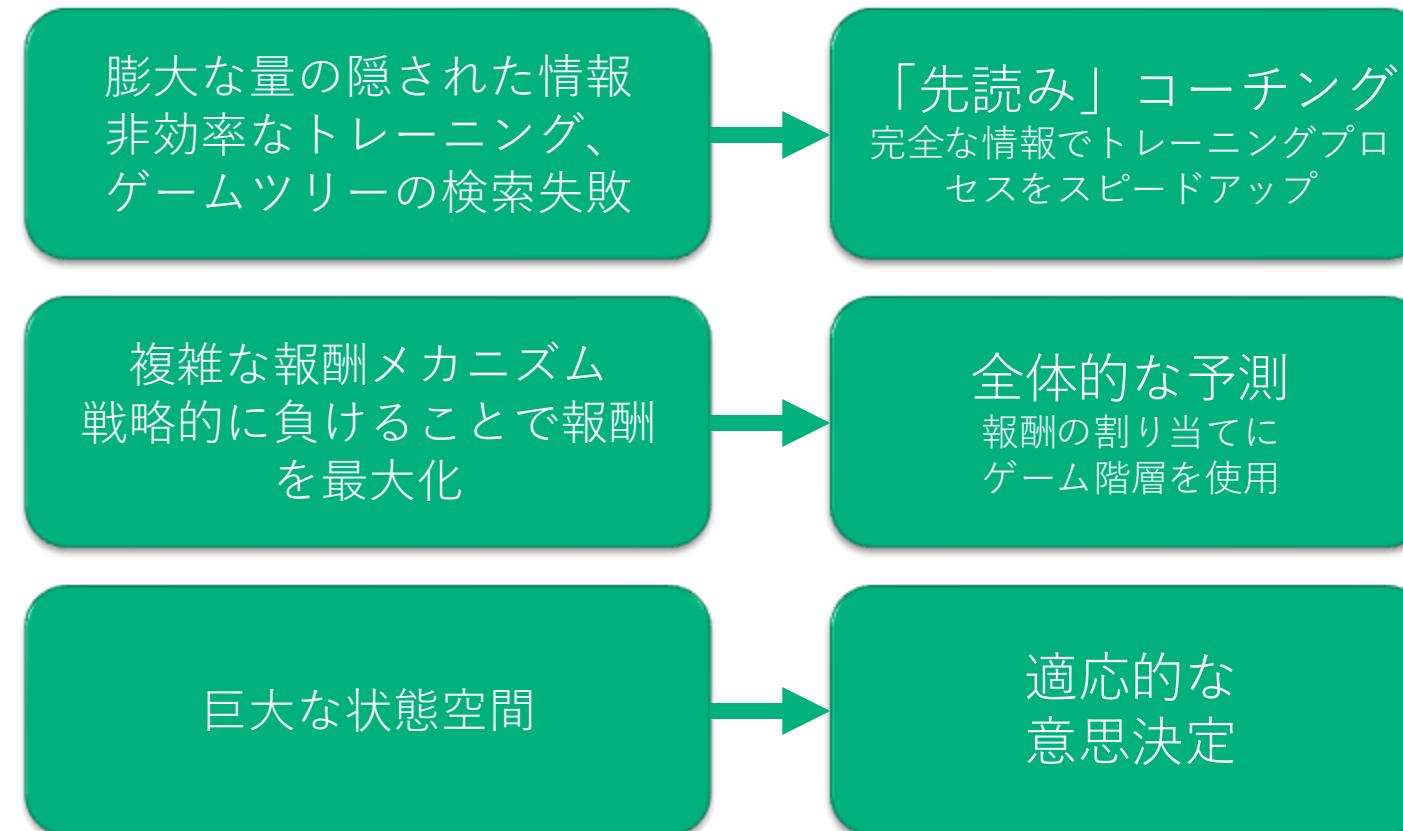
太くないお/藤井聰太/計算機自然
@Futokunaio_Sota

Suphx十段昇段おめでとうございます。牌譜は300以上は見てますがかなり上手いし参考にしています。
最近は人間の牌譜を見るのは辞めました。
僕は愛称をこめてすばふえにちゃんと呼んでいます。

午後8:46 · 2019年6月22日 · Twitter for iPhone

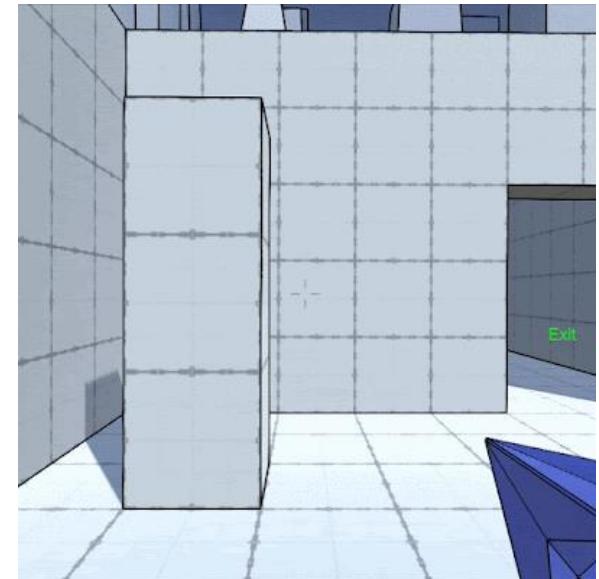
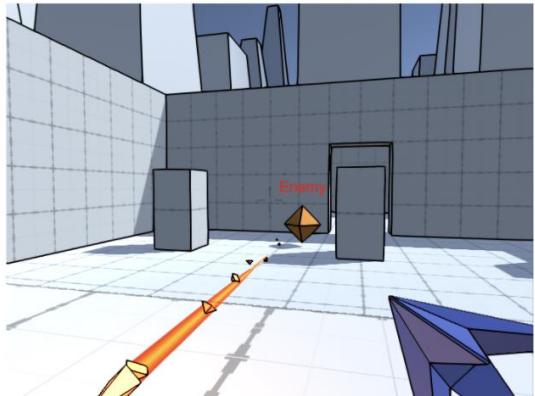
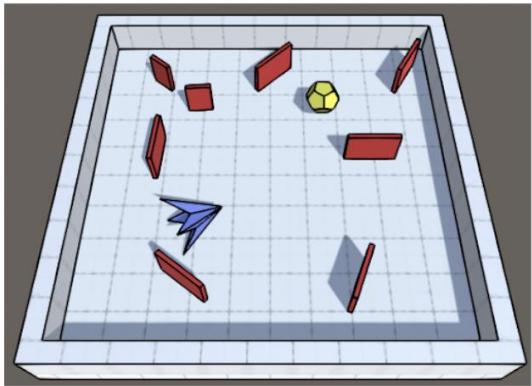
https://twitter.com/Futokunaio_Sota/status/1142398515588374528?s=20

- ・ 麻雀は「不完全情報ゲーム」の代表格
- ・ 不確実性の高い隠された情報が非常に多い
- ・ 優れた仮説、予測、推論、ファジーな意思決定能力が必要となる



Deep
Reinforcement
Learning

- 人間の本能、予測、推論、あいまいな意思決定能力、およびゲーム内の全体的な状況の感覚を示し、麻雀の高い不確実性に効果的に対処
- 人間の学習、麻雀のスキル向上、麻雀コミュニティの発展を支援
- 現実世界のシナリオ
 - 自動運転、金融投資など、ゲーム以外の分野のほとんどは未知の情報が多い
 - さらに、偶発的な要素にも影響を受けやすい
 - 世の中、社会の、複雑な問題の解決に貢献
- ただし、研究開発のコストは依然として大きい



AIの可能性

- ・ ゲームソースへのアクセス
- ・ ビデオゲームの対話的性質

RLのデメリットに着目

- ・ 固有のネットワークアーキテクチャ
- ・ MLアルゴリズムの実装の専門知識
- ・ 大量のトレーニングデータ

Falken

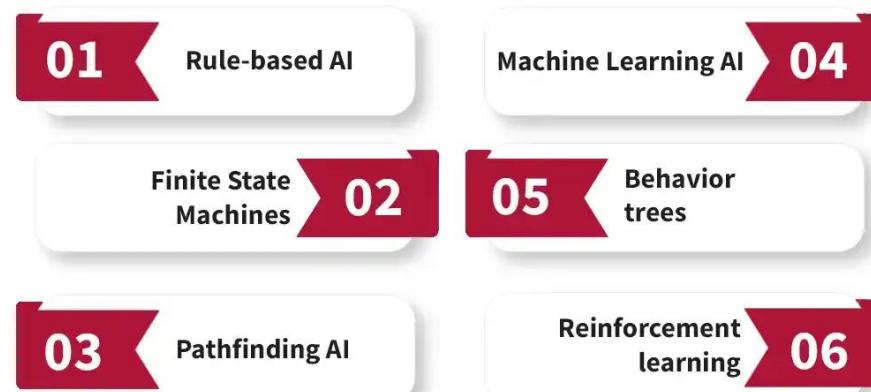
<https://github.com/google-research/falken>

ゲームをプレイできる AI をトレーニングできるサービスを Google がオープンソースにて提供
報酬やオフライントレーニングのバッチを通じて学習する従来の RL フレームワークとは異なり、
Falken はリアルタイムの人間との対話を介した AI のトレーニングに基づいている模倣学習です。

- ルールベースのAI
- 有限状態マシン
- パスファインディングAI
- 機械学習AI
- ビヘイビアツリー
- 強化学習



What are the types of AI in games?



<https://www.engati.com/blog/ai-in-gaming>

- ルールベースのAI
- 有限状態マシン
- パスファインディング
- 機械学習AI
- ビヘイビアツリー
- 強化学習

ルールベースのAI: 単純な条件分岐やルールに基づいて、敵の行動を制御

- 仕組み:
 - IF-THENルール: 条件を満たした場合に指定された行動を実行
 - 例: 「プレイヤーが視界内に入ったら攻撃する」
- 具体例:
 - Pac-Man (1980年) : ゴーストがプレイヤーを追跡
- 評価:
 - メリット: 実装が簡単で軽量
 - デメリット: プレイヤーに予測されやすい

- ルールベースのAI
- **有限状態マシン**
- パスファインディング
- 機械学習AI
- ビヘイビアツリー
- 強化学習

有限状態マシン (**Finite State Machine, FSM**)

敵の状態（待機、追跡、攻撃など）を明確に分け、条件に応じて状態遷移

仕組み：各状態で異なる行動を実行。条件が満たされると状態を遷移。

例：「待機」 → 「視界内にプレイヤー発見」 → 「追跡」。

具体例：The Legend of Zelda (1986年)

敵がプレイヤーを発見すると追跡状態に移行。

メリット：明確でシンプルな行動制御。

デメリット：状態数が多いと管理が煩雑。

- ルールベースのAI
- 有限状態マシン
- パスファインディングAI
- 機械学習
- ビヘイビア
- 強化学習

パスファインディングAI

- 障害物を回避しながら効率的な経路を計算するAI。
- 仕組み:
 - A*アルゴリズム: 最短経路を計算するための一般的な手法。
 - グリッドベースマップ: マップをグリッドに分割して移動可能なセルを評価。
- 具体例:
 - Doom (1993年) : 敵がプレイヤーの位置に基づいて最適なルートを計算し追跡。

- ルールベースのAI
- 有限状態マシン
- パスファインディングAI
- **機械学習AI**

機械学習AI

- **定義:** データを学習して行動パターンを見つけ、敵の動作に応用する技術。
- **仕組み:**
 - スーパーバイズドラーニング: ラベル付きデータで訓練。
 - アンラベルドラーニング: データから自律的にパターンを発見。
- **具体例:**
 - **F.E.A.R. (2005年)** : 敵がプレイヤーの行動に適応し、状況に応じてグレネード投擲やカバーを選択する。
 - **Forza Horizon (Turn 10 Studios)** : レースゲームで、プレイヤーの走行データを基にAI「Drivatar」が個々のプレイスタイルを学習し再現。
 - **Tom Clancy's Rainbow Six Siege (Ubisoft)** : AIがプレイヤーの戦術を学習し、防御や攻撃を適応的に変更。
- **評価:**
 - **メリット:** 過去データから合理的な意思決定を学習可能。
 - **デメリット:** データの偏りがAI動作に影響を与える。

- ルールベースのAI
- 有限状態マシン
- パスファインディングAI
- 機械学習AI
- **ビヘイビアツリー**
- 強化学習

ビヘイビアツリー (Behavior Tree)

- **定義:** 敵の行動を階層構造で管理し、状況に応じた適切な行動を選択する技術。
- **仕組み:**
 - **条件ノード:** 状況を評価し次のアクションを決定。
 - **アクションノード:** 実際の行動を実行。
- **具体例:**
 - **Haloシリーズ:** 敵が状況に応じて「カバーを取る」「突撃する」などを選択。
- **評価:**
 - **メリット:** 柔軟で複雑な行動を実現可能。
 - **デメリット:** ツリーが複雑になるとデバッグが困難。

- ルールベースのAI
- 有限状態マシン
- パスファインディ
- 機械学習AI
- ビヘイビアツリー
- 強化学習

強化学習: AIが試行錯誤を通じて最適な行動を学習する技術。

AIが試行錯誤を通じて報酬に基づいた行動を学ぶケースが典型です。

これらは大量のトレーニングデータと計算リソースを必要とするため、使用される場面が限られます。

仕組み: エージェント、環境、行動、報酬、方策を使う

具体例:

- **Gran Turismo Sophy:** 強化学習を用いてレーシングスキルを習得。プレイヤーと競争し、高度な戦術を実現。
- **AlphaStar (StarCraft II) :** 強化学習でプレイヤーの戦略を学習し、リアルタイムで適応するAI。
- **OpenAI Five (Dota 2) :** プロプレイヤーと競うための戦略を学習。

長所と短所:

- **長所:** 試行錯誤を通じて、未知の状況にも対応可能。
- **短所:** 学習コストが高く、膨大なリソースを必要とする。

- **ゲームバランスの崩壊**

- 強すぎ、または、弱すぎ
- AIに意図的にミスをさせる仕組みを導入
- 動的難易度調整

- **製作者の意図に合わない**

- AIが予想外の行動を取ることで、ゲームデザインの意図を損なう
- プレイヤーの体験を損ねる

- **リソース不均衡（過剰）**

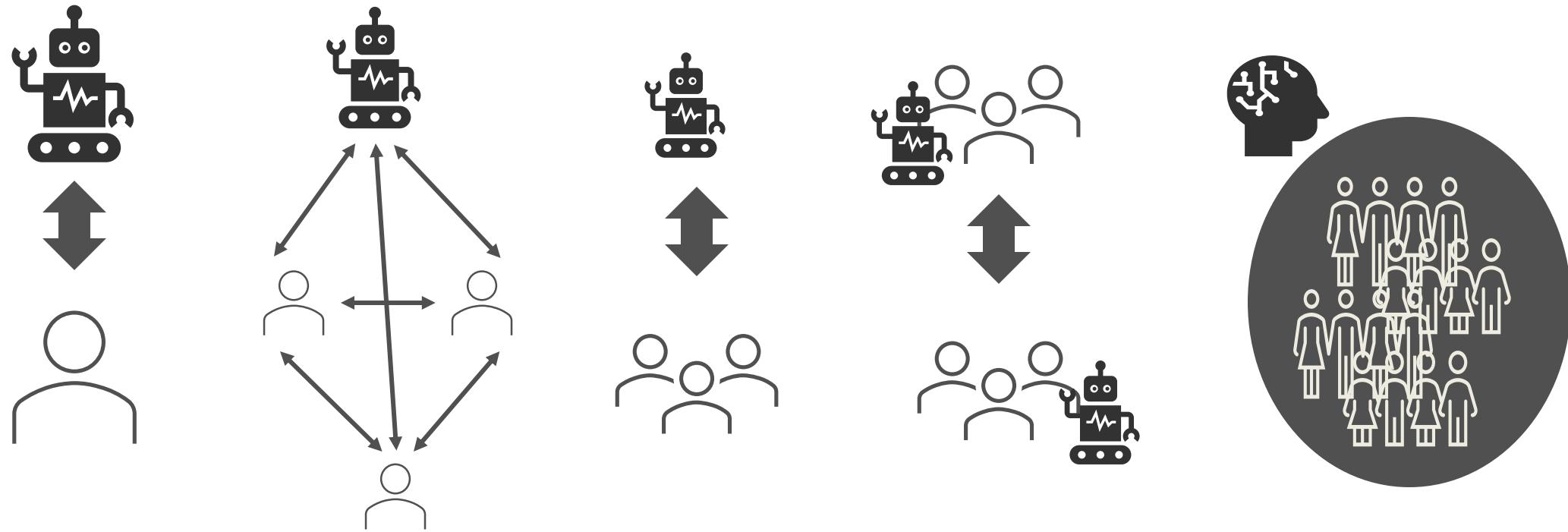
- ゲームシステムへの負担
- 複雑化
- コストの増加

- **プレイヤー行動の固定化**

- プレイヤーがAIの行動パターンを学習
- 特定の攻略法に依存することで、ゲームプレイが単調化。

- 人間対コンピュータのゲームの起源は1952年
 - ケンブリッジ大学院生による「OXO」
- ビデオゲームのAIは1970年代あたりから
 - ポン・スペースインベーダー・パックマンが、初期の導入事例
 - 初期のビデオゲームにおいて、高度なニューラルネットワークや機械学習は使用されず、基本的なアルゴリズムによる予測可能な行動パターンでプレイヤーとの対話を実現
- RL強化学習に注目 (Gameというより、GameをPlayするAI)
 - 大規模な試行錯誤、加速度的な学習プロセス、何千時間もの学習
 - OpenAI Dota 2、1万年以上の自己対戦で学習、ビデオ事前学習でさらに加速
 - Minecraftに、強化学習の微調整に加えて、「模倣学習」（人間の行動を見て学習するニューラルネットワーク）を応用するなど、新しいアプローチも多い

- ・ゲームの難易度、課題の複雑化、マルチプレイ化
- ・オンラインゲーム空間全体のバランスの調整など新たな課題



BLUE PROTOCOLのAI実装（CEDEC 2020 発表）



- パーティ VS パーティ
- プレイヤー集団の分析には「階層的クラスタリング」
- エネミーのパーティ形成は、警戒範囲内のプレイヤーをターゲットとして攻撃し、ターゲットリストを基に行動を決定します。
- エネミーの行動制御には、3つの階層
 - 情報収集の「Perception」
 - 意思決定の「Brain」
 - 実際の制御の「Action」
- 階層型タスクネットワークを用いて、タスクを細かく分割し、実行する
 - 「HTN Planning」アルゴリズム
 - タスク実行前後の制約を加えることで、エネミーが状況に応じて最適な行動を選択するように制御

The screenshot shows a news article from GameBusiness.jp. The header includes the site's logo, a search bar, and navigation links for Home, Game Development, Market, Management, Company Trends, Human Resource Development, Industry Policy, Collaboration, Events/Seminars, and Esports. The main content area features a dark blue banner with white text about the AI implementation in BLUE PROTOCOL. Below the banner is a large image of a presentation slide titled "BLUE PROTOCOL の AI 実装" with subtext "～企画意図をふるまいに反映させるために行ったこと～". The slide lists "株式会社 バンダイナムコスタジオ 佐竹 敏久" and "長谷 洋平 (共同研究者)". To the right of the slide is a sidebar for Shopify, which reads "shopify 24時間体制のサポートを活用してオンラインショップを運営 複雑なビジネス運営も、私たちがしっかりサポートします。 無料体験を始める". At the bottom of the page, there are two small images showing people interacting with products.

- Project Paidiaの発表: Microsoft Researchの強化学習 (RL)
- 協力ナビゲーションタスク
- 「Bleeding Edge」 (Ninja Theory) に採用
- エージェントが一連の決定を行うアプローチ
- 単一エージェントおよびマルチエージェントRL、オープンソース RLアルゴリズムへのアクセス、様々なゲーム環境への対応
- TensorFlow、ONNX、PyTorchモデルをゲーム内使用

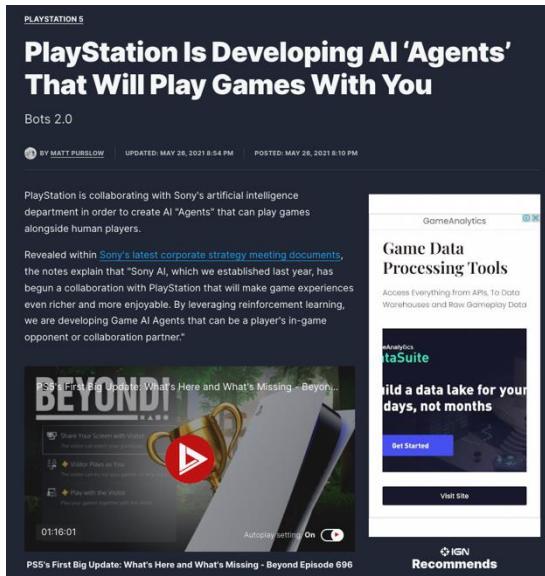


<https://developer.microsoft.com/en-us/games/articles/2020/08/supercharge-games-with-azure-ai-and-reinforcement-learning/>



Reinforcement Learning Advances With Project Paidia | IGL186
(3年前のIgniteにて公開)

<https://www.youtube.com/watch?v=rhnGNwTyA-8>



PlayStationはソニーの人工知能部門と協力して、人間のプレイヤーと共にゲームをプレイできるAI「エージェント」を開発中
エージェントは、プレイヤーのゲーム内の対戦相手や協力パートナーとなる

ゲームAIエージェントの特徴

- **特許技術の関連**
 - ソニーの最近の特許「ゲームアプリケーション中の特定タスクのための自動化された人工知能(AI)制御モード」に関連している可能性。
- **人間のプレイスタイルの模倣**
 - 人間のユーザーのプレイスタイルに基づいて「プレイをシミュレート」するAI。
- ゲーム内AIキャラクターの進化
- エージェントは、協力プレイやPvPゲームでの人間のプレイヤーに近い存在として設計されている。
- **マルチプレイヤーゲームにおける「ボット」への適用**

<https://www.ign.com/articles/playstation-game-ai-agents>

- ・ゲーム開発とAI の概要
- ・ゲームにおけるAI利用の変遷と拡張
- ・生成AIはゲーム開発に使えるの？
- ・ゲーム産業における象徴的なAI事例
- ・まとめと振り返り

ゲーム業界の歴史 (1970-2020)

1970-1983: ゲームクラッシュの前夜

- 初期のビデオゲーム「Pong」
- アーケード → 家庭用「Pong」（1975）
- Atari 2600（1977）が100万台以上を売上げ

1985-2000: 技術進歩競争

- AtariはPac-ManとE.T.のゲームで失敗
- 任天堂のNESは、高品質とマーケティングで勝利
- セガ、パナソニック、ソニー、マイクロソフト

2001-現在: オンラインブーム

- インターネットとモバイルの台頭
- 業界は数十億ドルから数百億ドル規模に成長
- MicrosoftはXbox Liveを開始
- BlizzardはWorld of WarcraftでMMO市場を開拓
- Appleがモバイルプラットフォームのゲームを確立

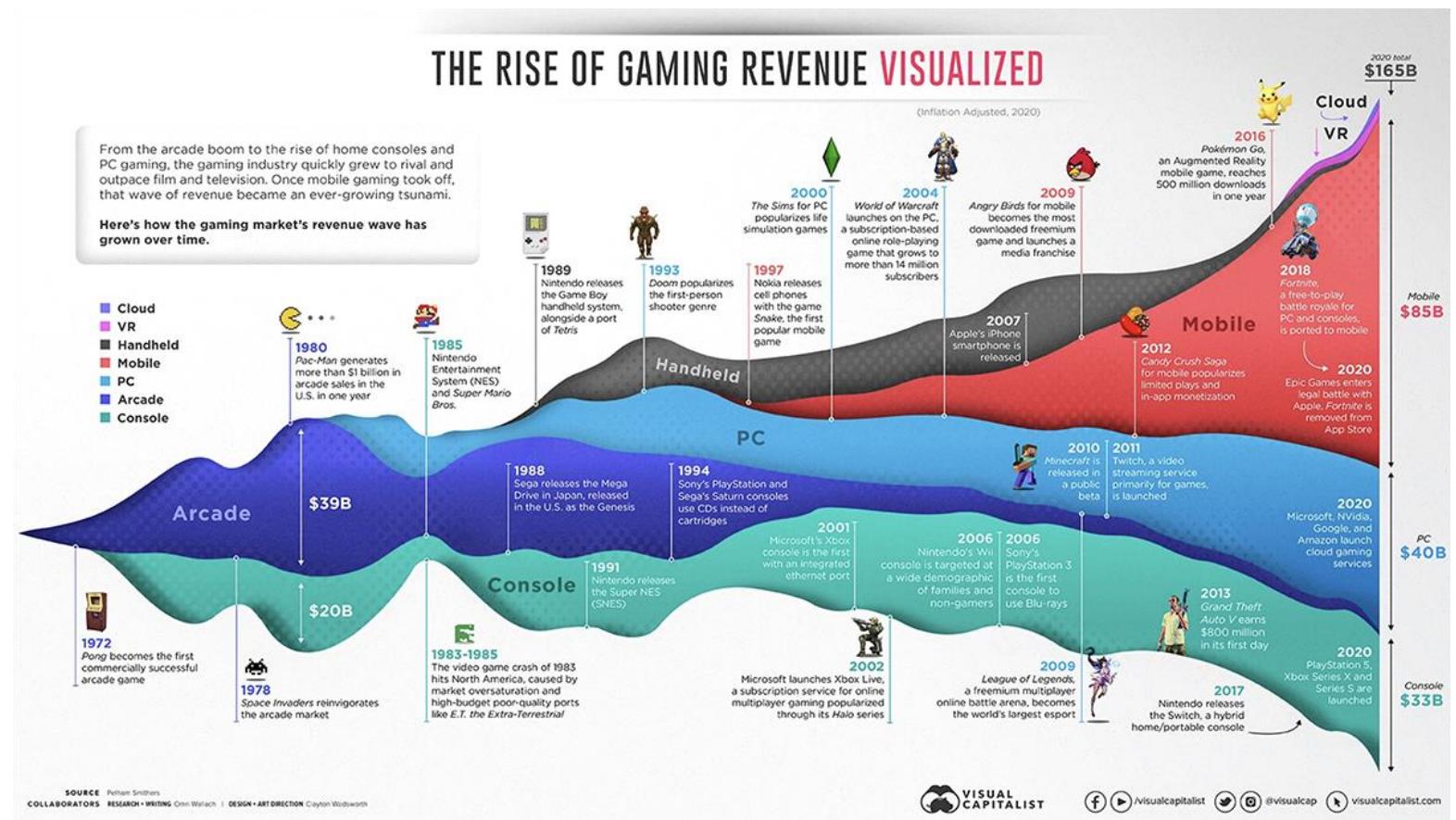
主要なゲーム買収（2014年以降）

- Facebook : Oculus (VR) 30億ドル
- Amazon : Twitch (ストリーミング) 10億ドル
- Microsoft : Mojang (ゲーム) 25億ドル

最近のトレンド

- MicrosoftやSonyは新しいコンソールを開発
- 各社は、クラウドベースのサブスクリプション提供
- ダウンロードゲーム + クラウドゲーミング
- AIの高度な活用

2020年時点で、世界中に27億人以上のゲーマーが存在
彼らの支出方法が今後のゲーム業界を形作る



50 Years of Gaming History, by Revenue Stream (1970-2020)

<https://www.visualcapitalist.com/50-years-gaming-history-revenue-stream/>

特徴:

- 簡単なルールベースのアルゴリズムを利用した敵の行動ロジック。ルールベースアルゴリズム
- 計算リソースが限られていたため、シンプルながらも戦略的な行動を実現。

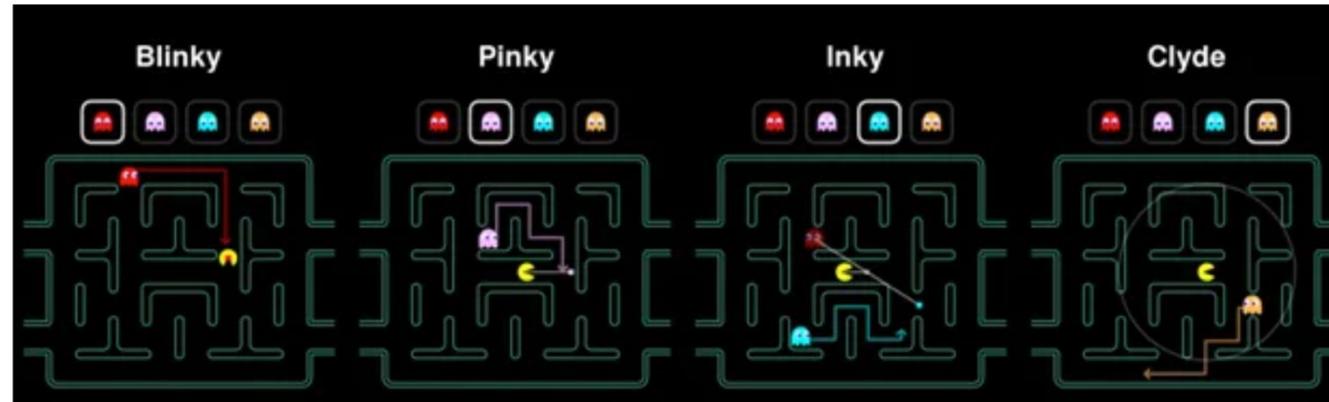
代表作品と仕組み:

1. Space Invaders (1978年)

- 敵キャラクターが一斉に動き、プレイヤーに迫る。

2. Pac-Man (1980年)

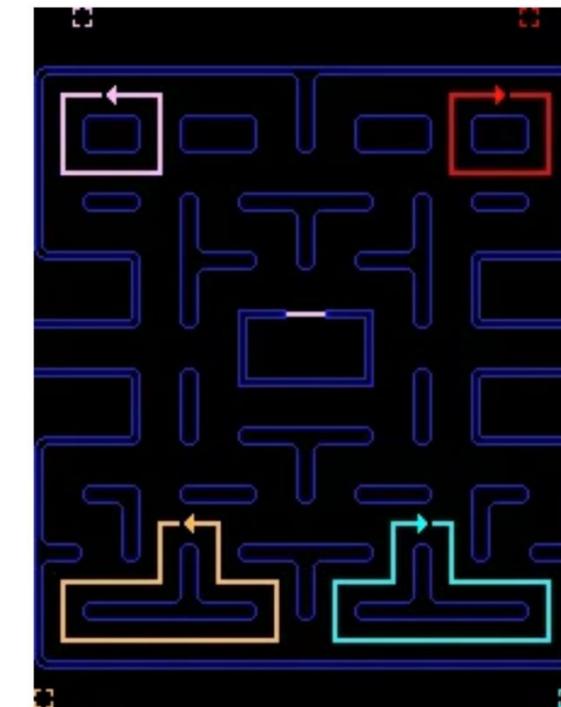
- ゴースト（敵キャラクター）が異なるパターンでプレイヤーを追跡。



Pac-Man with OOP

<https://medium.com/@shivangk1407/pac-man-with-oop-1a26ae6c3c87>

Chase Mode Visualized



Scatter Mode Visualised

特徴:

- ・ アーケードゲームや家庭用ゲーム機の性能向上により、敵AIがより複雑化。
- ・ 状態遷移（State Machine）を用いた敵の行動制御が一般的に。

代表作品と仕組み:

1. The Legend of Zelda (1986年)

- 敵が特定の条件下でプレイヤーを追跡または攻撃。
- 仕組み: 敵が「追跡モード」と「待機モード」を切り替える状態遷移を採用。

2. Super Mario Bros. (1985年)

- 敵が単純な動きをしながら、プレイヤーの行動に応じて反応。
- 仕組み: 画面内に入った敵だけがアクティブ化される省メモリ設計。

3. Doom (1993年)

- FPSジャンルで敵がプレイヤーを見つけて攻撃する仕組みを導入。
- 仕組み:
 - ・ ラインオブサイト（視線）の概念を利用し、敵がプレイヤーを検知。
 - ・ パスファインディングでプレイヤーを追跡。

特徴:

- ・ 敵AIが環境やプレイヤーの行動に適応可能に。
- ・ チームAIや連携行動が重要視され戦略的な挑戦の機会を提供。

代表作品と仕組み:

1. Halo: Combat Evolved (2001年)

- 敵がグループで戦術的に連携。プレイヤーの攻撃に応じて位置を変える。
- 仕組み:
 - ・ 状態遷移を高度化し、カバーを取る、退避する、攻撃するなどを選択。
 - ・ 敵の個性を出すために種族ごとに行動パターンを変更。

2. Half-Life (1998年)

- NPC兵士がプレイヤーの位置に基づいて戦略的に行動。
- 仕組み:
 - ・ パスファインディングにA*アルゴリズムを採用。
 - ・ グレネードを使った攻撃やプレイヤーの位置を推測した動き。

A*
Wikipediaより

3. F.E.A.R. (2005年)

- 敵がカバーを取り、退避し、仲間と連携してプレイヤーを攻撃。
- 仕組み:
 - ・ プランニングAIを使用し、敵が目標を達成するために複数の行動を組み合わせる。

特徴:

- ・ 強化学習や機械学習を取り入れた敵AIが登場。
- ・ プレイヤーの行動に適応し、ダイナミックな挑戦を提供。

代表作品と仕組み:

1. The Last of Us Part II (2020年)

- 敵AIが連携し、プレイヤーを追い詰める。感情的な反応を行動に反映。
- **仕組み:**
 - ・ 状態遷移を用いながらも、動的なチームAIで個体間の連携を強化。
 - ・ プレイヤーが見つからない場合でも、音や視覚の情報を頼りに推測して行動。

2. Middle-earth: Shadow of Mordor (2014年)

- Nemesisシステムにより、敵がプレイヤーの行動を記憶し、再戦時に個別の因縁を形成。
- **仕組み:**
 - ・ 敵ごとに固有のデータベースを持ち、行動履歴を保存。
 - ・ プレイヤーとの関係性を動的に変更。

AIのゲーム業界での利用は1960～70年代に始まる。基本は対戦相手（エネミー）の演出

シンプルなルールベースAI

- 例：Pac-Man: ゴーストの行動パターンがルールに基づき設計され、戦略的なプレイを実現。
- 例：チェスAI: プレイヤーと対等に戦う意思決定ロジックを搭載。



「敵」ではなく「ゲームそのものを開発」するAI

AIは、ゲーム開発プロセスの効率化を実現する重要なツール。プロシージャル生成技術を用いることで、広大なマップや多彩なキャラクターを短期間で作成可能になる。また、AIを活用した自動テストにより、開発者の負担を軽減する。



AIでのマップ生成



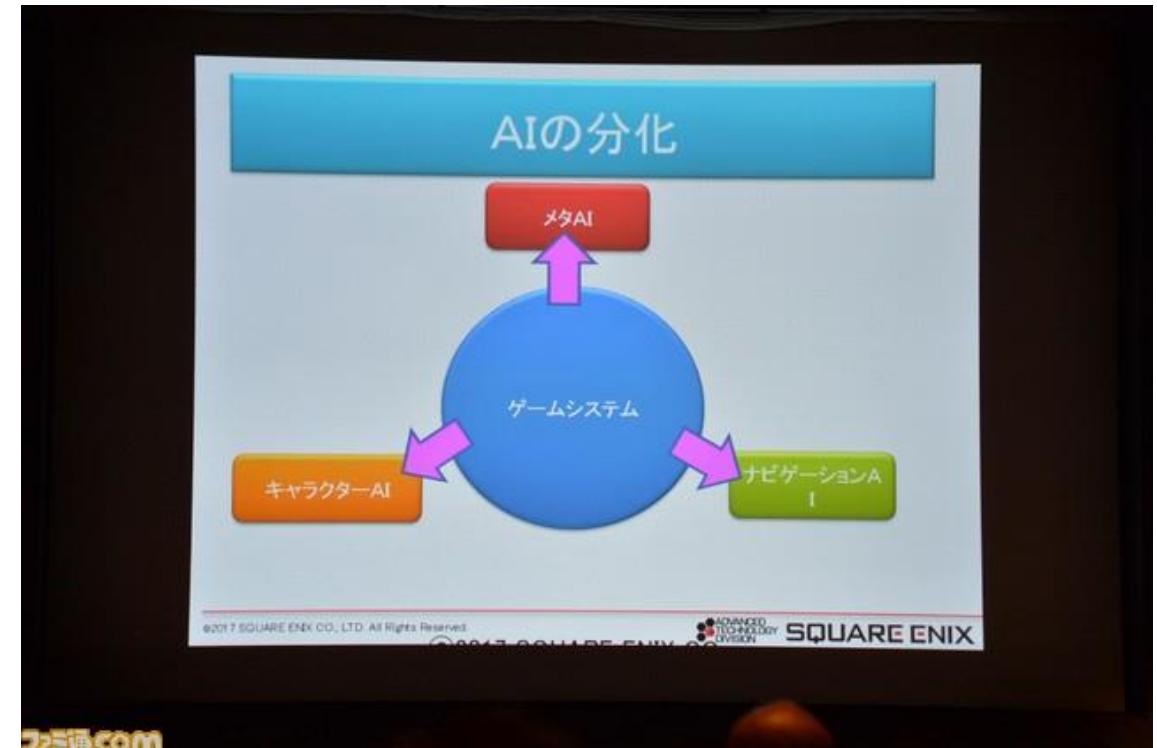
テストプレイの自動化

AIは、プレイヤーの選択や行動に応じてゲーム体験をカスタマイズし、没入感を向上させる。→ 適応型AIを使えば、
プレイヤーのスキルレベルに応じた難易度調整が可能になる

NPCの行動をリアルにすることで、
ゲーム世界の「生きた」感覚を提供

「The Last of Us」の例: 敵AIがプレイヤーの位置や行動に応じて戦術を変更。

「AI Dungeon」の例: AIがリアルタイムでストーリーを生成し、プレイヤーの選択に応じて物語を展開。



スクウェア・エニックスの研究者が提案する、「メタAI」

ゲームに組み込まれるもの

- In-Game AI
 - 敵キャラクター、NPC（非プレイヤーキャラクター）、動物、環境要素など、ゲーム内でプレイヤーと直接関わる要素を制御します。
 - わかりやすい実感しやすいAI
- Out-Game AI
 - ゲーム全体の進行を監視し、神的な視点で全体に影響するもの、補助的なもの
 - バックグラウンドで動作する、バランス調整、マッチメイキング、報酬など

ゲームに組み込まれないがゲーム制作に大きな影響を与えるAI

- クリエイティブAI（制作作業におけるAI）
 - コンテンツ生成やデザイン支援
- テストAI
 - 自動テスト、テストデータ分析、動的デバッグ

エネミーAI 以外にもさまざまな利用が進む



カテゴリ	In-Game AI	Out-Game AI	クリエイティブ/テストAI
役割	ゲーム内のキャラクターや環境を制御	プレイヤー体験を調整	ゲーム制作プロセスを効率化・自動化
使用目的	没入感の向上	体験の最適化	制作コストの削減、品質向上、効率化
動作の範囲	ゲーム内	ゲーム外	制作ツールやQAプロセスの一部
例	敵の行動、NPCの対話	マッチメイキング、難易度調整	レベルデザイン生成、QA自動化、ストレステスト



これらのAIは、将来、連携、統合される可能性がある

また、生成AI（Generative AI）の台頭により、個人作業にもAIが浸透
ますますAIがゲーム制作に大きな影響を与える

ゲームプレイ中に動作し、プレイヤーの体験を直接向上させるAI。

ジャンル名	定義	代表例	技術の特徴
ダイナミックAI	プレイヤーの行動や状況に応じて、NPCやゲーム環境がリアルタイムで適応し、変化する技術。	- The Last of Us Part II : 敵が連携してプレイヤーに対応 - Left 4 Dead : ゾンビの出現を調整	- 状態認識と行動生成 - 強化学習 (Reinforcement Learning) - プレイヤーモデリング
適応型AI	プレイヤーのスキルやプレイスタイルに基づき、難易度やゲーム展開を動的に調整する技術。	- Hades : 難易度をスキルに応じて調整 - Resident Evil 4 : 敵配置やアイテムドロップを変更	- プレイヤーモデリング - 動的難易度調整 (Dynamic Difficulty Adjustment)
ソーシャルAI	NPCがプレイヤーや他のNPCと自然な会話や感情的な反応を行う技術。	- Cyberpunk 2077 : NPCがリアルな日常行動をシミュレーション - Inworld AI : NPCが動的な会話を生成	- 自然言語処理 (NLP) - 感情シミュレーション - 音声合成 (TTS)
シミュレーションAI	環境や生態系の動きをリアルに再現し、プレイヤーがその影響を体験できる技術。	- SimCityシリーズ : 都市環境のシミュレーション - Red Dead Redemption 2 : リアルな動植物の行動	- エージェントベースモデリング - パーティクルシステム - ルールベースモデル

ゲーム外でプレイヤーや開発者に価値を提供するAI。

ジャンル名	定義	代表例	技術の特徴
マッチメイキングAI	プレイヤーのスキルやプレイスタイルを分析し、公平で楽しめる対戦相手やチームを自動的に選定する技術。	- Apex Legends : 実力に基づく対戦マッチング - Overwatch : ランク別マッチメイク	- クラスタリング - レコメンデーションシステム
スコアとリーダーボードAI	プレイヤーのパフォーマンスを分析し、ランキングを生成する技術。	- Fortnite : グローバルリーダーボード - Call of Duty : 個人スコアと統計の追跡	- 統計分析 - データ可視化モデル
ソーシャルゲームAI	ゲーム内のフレンド機能、チャット、ギルド管理など、プレイヤー同士の交流を支援する技術。	- Clash of Clans : ギルドシステムとチャット - Roblox : ソーシャル機能と協力プレイ	- 自然言語処理 (NLP) - ネットワーク解析
ユーザー行動解析AI	プレイヤーの行動データを収集・分析し、ゲームバランスの改善やパーソナライズされた体験を提供。	- Netflixのゲーム推奨システム - モバイルゲーム広告最適化	- 機械学習 (クラスタリング、分類) - 行動予測モデル
カスタマーサポートAI	プレイヤーの質問や不具合報告に迅速に対応し、サポートコストを削減。	- EAのサポートAI : プレイヤーの質問を自然言語処理で解釈し、自動回答。	- 自然言語処理 (NLP) - FAQ生成

開発者を支援し、制作やテストの効率を向上させるAI。In-Game/Out-Game に負けずかなり注目されている

ジャンル名	定義	代表例	技術の特徴
プロシージャルAI	規則やアルゴリズムに基づいて、ゲーム内コンテンツ（地形、敵、アイテムなど）を自動生成する技術。	<ul style="list-style-type: none"> - No Man's Sky: 惑星や生態系をリアルタイムで生成 - Minecraft: 地形や構造物のランダム生成 	<ul style="list-style-type: none"> - 擬似乱数生成 (PRNG) - フラクタル生成 (Perlin Noise) - ルールベース生成
テスト支援AI	自動でゲームの不具合を検出し、開発者にフィードバックを提供。	<ul style="list-style-type: none"> - GameDriver: 自動プレイテスト - Keywords StudiosのAI QA: バグ検出 	<ul style="list-style-type: none"> - パターンマッチング - 強化学習 (AIがテストプレイを学習)
アート生成AI	ゲームのテクスチャ、キャラクターデザイン、エフェクトを効率的に作成。	<ul style="list-style-type: none"> - Unity ArtEngine: テクスチャ自動生成 - MidJourney: コンセプトアート作成 	<ul style="list-style-type: none"> - GAN (Generative Adversarial Networks) - スタイル変換モデル

- ゲーム開発とAI の概要
- ゲームにおけるAI利用の変遷と拡張
- 生成AIはゲーム開発に使えるの？
- ゲーム産業における象徴的なAI事例
- まとめと振り返り

パックマンゴーストAIの本質

- ・ パックマンの各ゴーストは、見かけはシンプルですが、以下の重要な特徴を持っています：
- ・ 予測可能性：プレイヤーが学習できる一貫した行動パターン
- ・ 個性：各ゴーストの独自の動きによる戦略性
- ・ 即時性：ミリ秒単位での素早い判断と行動
- ・ 効率性：最小限のリソースで動作する軽量な処理

2. 汎用LLMの限界

- ・ 汎用LLMをそのまま使用する場合の問題点：
 - ・ レイテンシ：APIコールに数百ミリ秒が必要
 - ・ 一貫性：同じ入力でも異なる出力が生成される
 - ・ リソース効率：常時接続と高いコンピューティングリソースが必要
 - ・ コスト：API使用料が累積する
- ・ **現実的な活用方法** (LMの効果的な活用シーン)
 - ・ 開発時のNPC会話文生成
 - ・ ゲームそのものの企画やプログラムコードを生成
 - ・ シナリオやクエストの自動生成
 - ・ プレイヤーの行動パターン分析
 - ・ QAテスト時の自動プレイ

- 注目は大きい、本番適用は慎重
- LLM + マルチモーダル (VLMなど) の技術発展の組み合わせを模索
- ビジュアル情報や音声情報と組み合わせた新たな体験や開発フローが生まれつつある
- 大規模な開発現場では、高度なカスタマイズや業務要件への適合が不可欠
 - 生の生成AIサービス（クラウド+プロンプト+RAG）は限界がある
- 特定のシナリオでは、かなり利用が進んでいる
 - 企画作り（ストーリー、キャラクター）、絵素材のラフ作成、教育分野、など

In-Game AI 主な用途:

- NPCとの自然な対話システム
- クエスト・物語の自動生成
- シナリオの動的調整

開発支援 主な機能:

- シナリオ・台詞の自動生成
- ステージ設計の分析と改善
- バグ検出の自動化

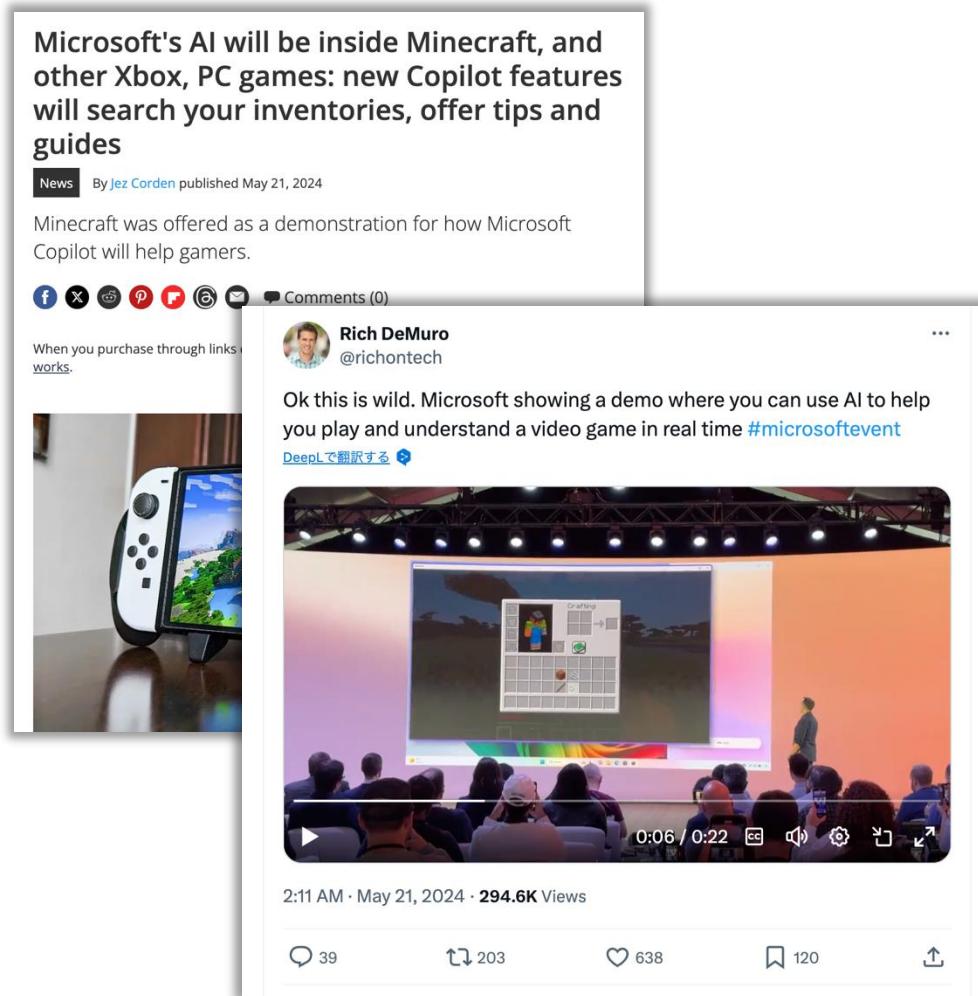
Out-Game AI 主な用途:

- プレイヤーデータの解析と最適化
- カスタマーサポート自動化
- コミュニティ管理

将来展望：業界特化アプリの開発

- 視覚情報処理の統合
- 多様なインタラクション対応
- 企業固有の要件対応

事例：Microsoft to integrate AI into games



1. 主な機能

Minecraftでの実装例

- ・自然言語による質問機能
- ・インベントリ内の自動検索
- ・クラフトレシピのガイド
- ・必要素材の入手方法案内

特徴

1. ゲーム内で直接質問可能
2. リアルタイムの状況に応じた回答
3. Alt+Tabでの検索が不要に

2. 技術的特徴

- ・OpenAIのChatGPTとDalle-3言語モデルを活用
- ・デバイス内でのデータ処理
- ・プライバシー重視の設計

3. 想定される影響

- ・初心者の学習障壁低下
- ・ゲーム体験の効率化
- ・シームレスなゲームプレイ

- テキスト生成
 - ChatGPT/Copilot
 - シナリオ、キャラクター設定
 - コード生成、デバッグ支援
 - ビジュアル生成
 - Midjourney/DALL-E/Stable Diffusion
 - コンセプトアート
 - キャラクターデザイン
 - 背景、テクスチャ
 - オーディオ生成
 - Voicevox: キャラクター voice
 - Suno AI: BGM、効果音
 - 動画生成
 - Lumiere: プロモーション映像
- ※注意点
- 著作権・ライセンスの確認
 - 生成物の品質チェック必須
 - あくまで補助ツール

- 生成AIの得意分野の一つに、プログラミング、企画作成、があります
- AIはゲーム制作を「学ぶ」と「作る」ことの両方同時に貢献
 - 「簡単に作れる」ことが初心者を引き込む
 - 「基本知識を得やすい」環境が継続的な成長を可能にする
- 将来的には、AIが提供する「制作ツール」と「学習支援」がさらに統合され、初心者でも高度なゲーム制作が行える時代が加速すると考えられます

要素	簡単にゲームが作れる時代	ゲーム制作の基本知識を得やすい
対象	アイデアを形にしたいクリエイター	学びたい初心者
目的	実際のゲーム制作とリリース	学習とスキル向上
使用するAIの役割	制作プロセスの効率化、自動化 企画レベルのゲームを作成代行	メカニクスや、技術用語を説明 図解などを用いた、可視化

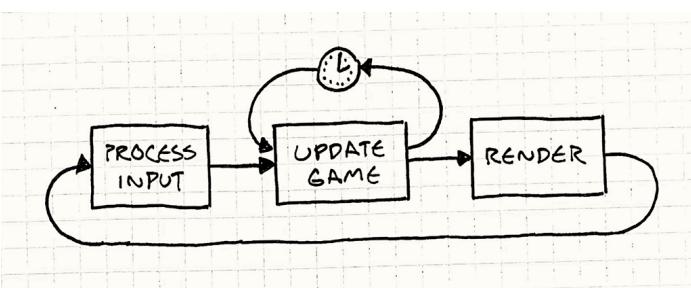
- **AIの勉強**になる
ゲーム開発の流れを通して、画像生成やテキスト生成などのAI技術を試せる
- **ゲームの勉強**にも役立つ
メインループ・描画処理など基礎構造を手を動かして理解できる
- **プログラムを書きたくない！？ プロデューサー志望でもOK**
「AI」のおかげで、簡単にプログラミングを行いプロトタイプを作れる
- **少人数・個人でも挑戦できる**
アセット作成（ガチャチケなど）をAIに手伝わせればゲームを実現しやすい

人が生み出すアイディア + AIの力
で 「個人ゲーム」 を作ってみましょう！

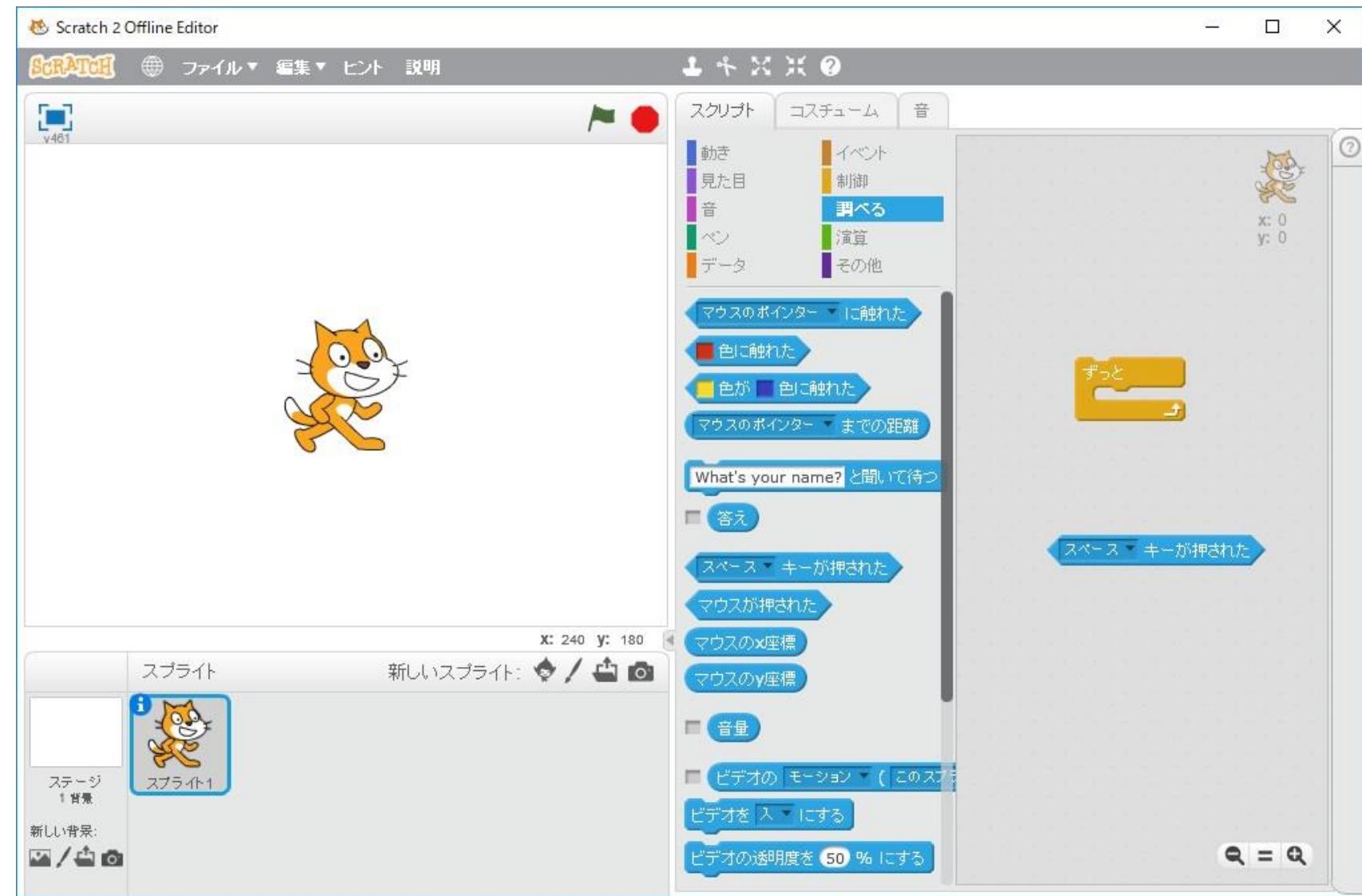
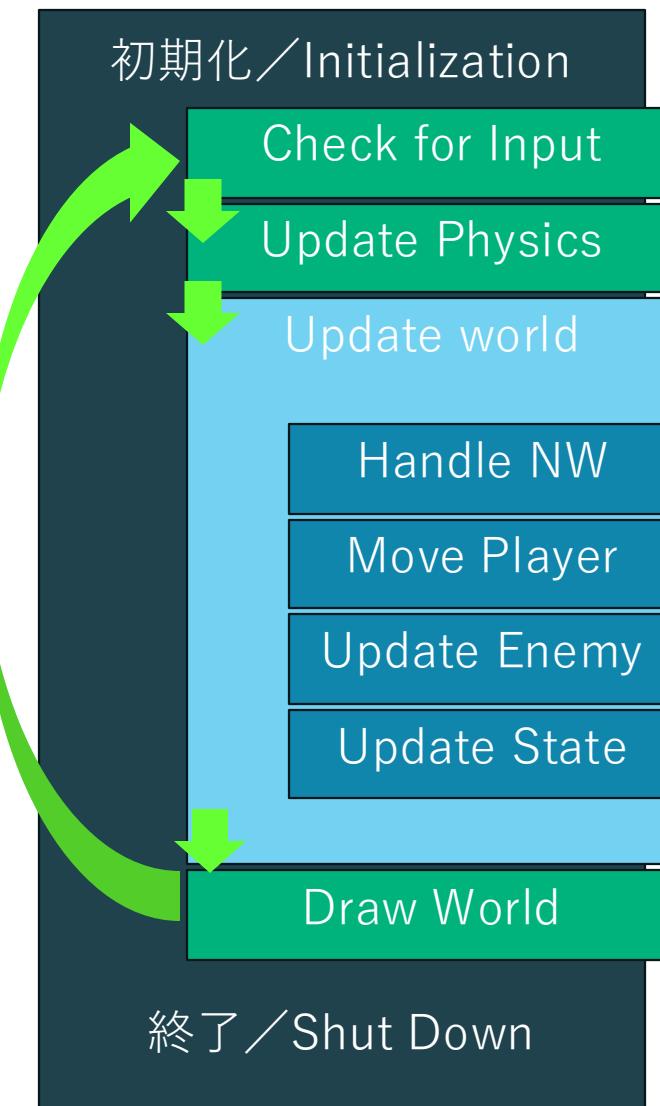
```
while (true){  
    processInput();  
    update();  
    render();  
}
```

ゲームループ、フレームレート、一貫したゲーム体験

- ゲームループ
 - ゲームの基本動作を制御する連続的な処理
 - ユーザーの入力処理、画面の描画処理を行う
 - ゲームのロジック（キャラクターの動き、スコア計算）を含む
- フレームレート、FPS、Frames Per Second
 - 1秒間に画面が更新される回数
 - 一般的なフレームレートは、30FPSや60FPSなど
- ハードウェアの違いを意識しない体験
 - 補間 (interpolation) や外挿 (extrapolation)
 - グラフィックス設定やフレームレート制約
 - 固定タイムステップ（ゲームロジック処理の更新を 60回／秒にする）



ループでできている



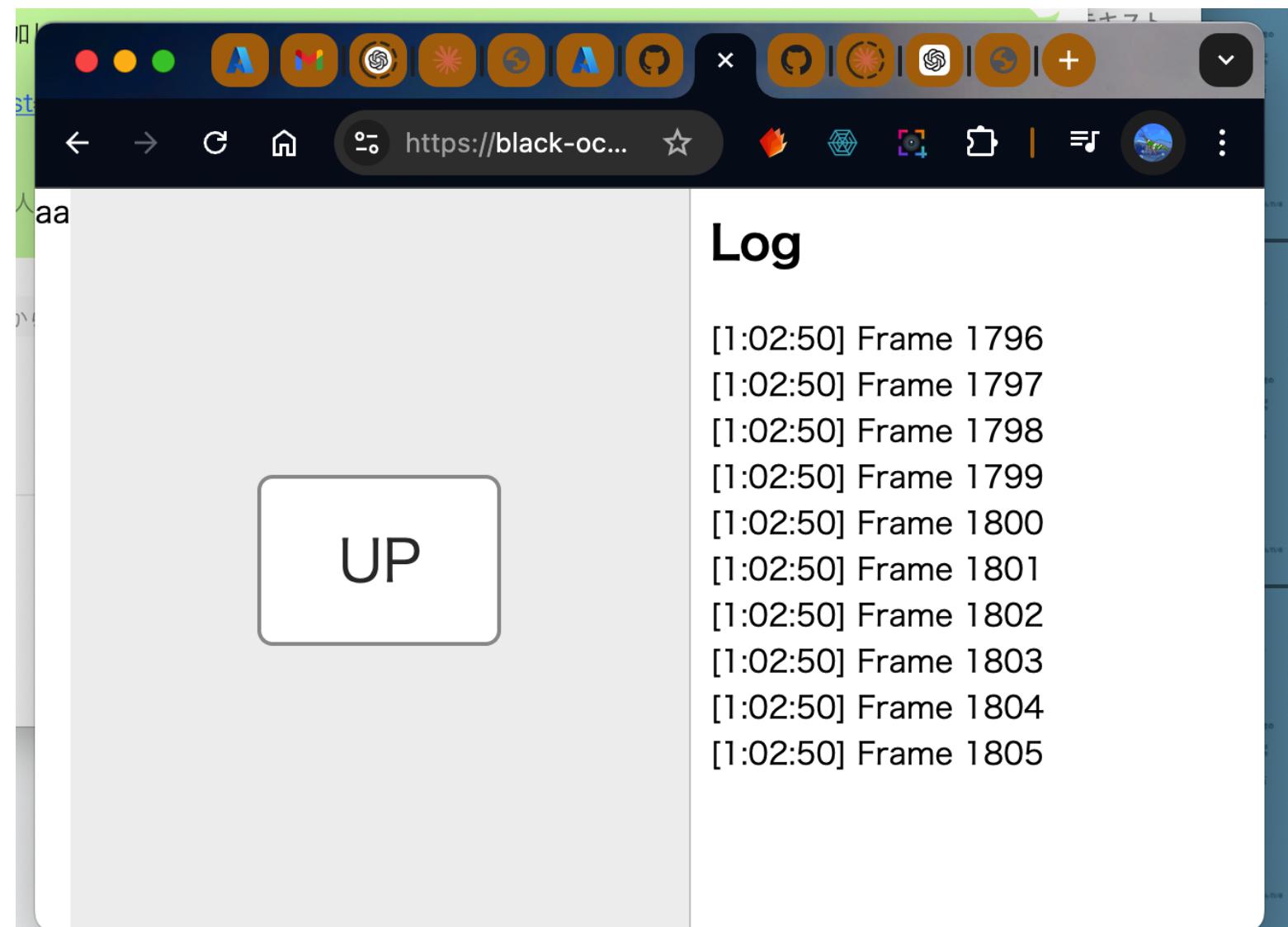
Sample Game



The image shows two side-by-side screenshots. On the left is a GitHub repository page for 'dmasubuchi / KITGames'. The repository has 87 stars and 1 fork. It contains files like .github/workflows, JankenLogger, MazeGame10Maps, Phyx2DJump, keyinputOperation, myFruitGame1, myFruitGame2, .DS_Store, Dockerfile, README.md, index.html, and staticwebapp.config.json. A 'Languages' section shows a chart with JavaScript at 69.8%, HTML at 24.1%, CSS at 5.6%, and Dockerfile at 0.5%. The README file contains the text 'KurumeGameAI , GameAI-Workshop'. On the right is a web page titled 'Kurume-KIT Game AI Collection' featuring several game prototypes: 'キー操作実験' (Arrow key experiment), 'フルーツキャッチ 1' (Fruit Catch 1), 'フルーツキャッチ 2' (Fruit Catch 2), '2D物理ジャンプ' (2D Physics Jump), 'じゃんけんロガーゲーム' (Rock-Paper-Scissors Log Game), and '迷路ゲーム (10Maps)' (Maze Game (10Maps)). Each prototype has a 'ゲームを開く' (Open Game) button.

ソース : <https://github.com/dmasubuchi/KITGames/tree/main>

デモ : <https://black-ocean-0fc6c5310.4.azurestaticapps.net/>



主要な機能

1. キー入力検出

1. 矢印キー（上下左右）の入力を監視
2. 入力されたキーを画面中央に表示
3. ログにキー入力を記録

2. ログ管理

1. 最新10件のログを保持
2. タイムスタンプ付きで表示
3. 自動スクロールで最新を表示

1. 初期化・変数設定

```
const arrowDisplay = document.getElementById("arrow-display");
const logContainer = document.getElementById("log-container");
let frameCount = 0;
let logLines = [];
```

2. キー入力処理

```
document.addEventListener("keydown", (e) => {
  // 矢印キーの判定
  // 画面更新とログ追加
});
```

3. メインループ (gameLoop)

```
function gameLoop() {
  frameCount++;
  addLog(`Frame ${frameCount}`);
  requestAnimationFrame(gameLoop);
}
```

4. ログ管理システム

```
function addLog(message) {
  // タイムスタンプ付加
  // 10件を超えたたら古いものを削除
  // 画面更新
}
```

• FPS (フレームレート) とは

- 動画やゲームの画面は、静止画を連続して表示することで動きを表現しています。この1秒間に表示される静止画の枚数を**FPS (Frames Per Second)** またはフレームレートと呼びます。FPSの値が大きいほど、画面の動きは滑らかになります。
- 人間の目は、ある程度のフレームレート以上であれば動きを滑らかに感じ取ることができます。一般的に、30FPS程度で滑らかな動画として認識され、60FPSではさらに滑らかで、動きの速いゲームなどに向いています。

• requestAnimationFrame

- `requestAnimationFrame` は、ウェブブラウザに搭載されているAPIの一つです。ブラウザに次のフレームを描画するタイミングで、指定した関数を呼び出すよう指示することができます。
- ゲーム開発において `requestAnimationFrame` を使用すると、画面のリフレッシュレートと同期して処理を実行できるため、滑らかなアニメーションを実現し、CPUやGPUの負荷を軽減することができます。

• ゲームループ

- ゲームは、内部で状態の更新、描画、入力処理などを繰り返すことで進行します。この処理の流れを**ゲームループ**と呼びます。
- `requestAnimationFrame` を用いることで、ゲームループをブラウザの画面更新と同期させることができます。

JavaScript



```
function gameLoop() {
    // 1. 状態の更新（キャラクターの位置、敵の行動など）
    // 2. 描画（更新された状態に基づいて画面を描画）
    // 3. 入力処理（プレイヤーからの入力を受け付ける）

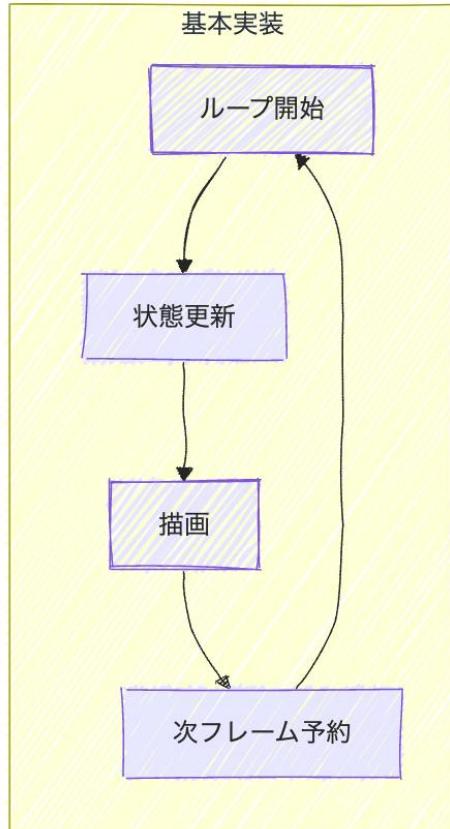
    // 次のフレームで gameLoop を実行
    requestAnimationFrame(gameLoop);
}

// ゲームループを開始
requestAnimationFrame(gameLoop);
```

60FPSの場合

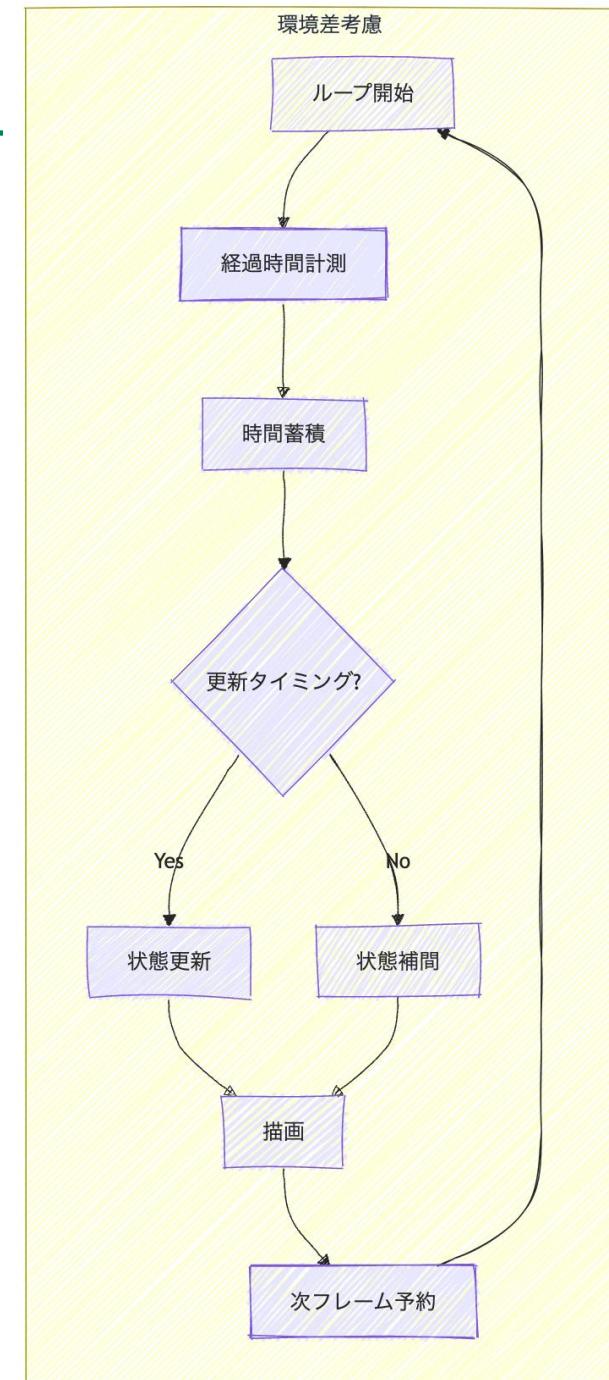
画面が60FPSの場合、`requestAnimationFrame` は1秒間に約60回 `gameLoop` 関数を呼び出します。

将来・商用環境では、遅延を考慮



稼働環境ごとの性能、
さまざまな遅延時間を
考慮してループを再構築

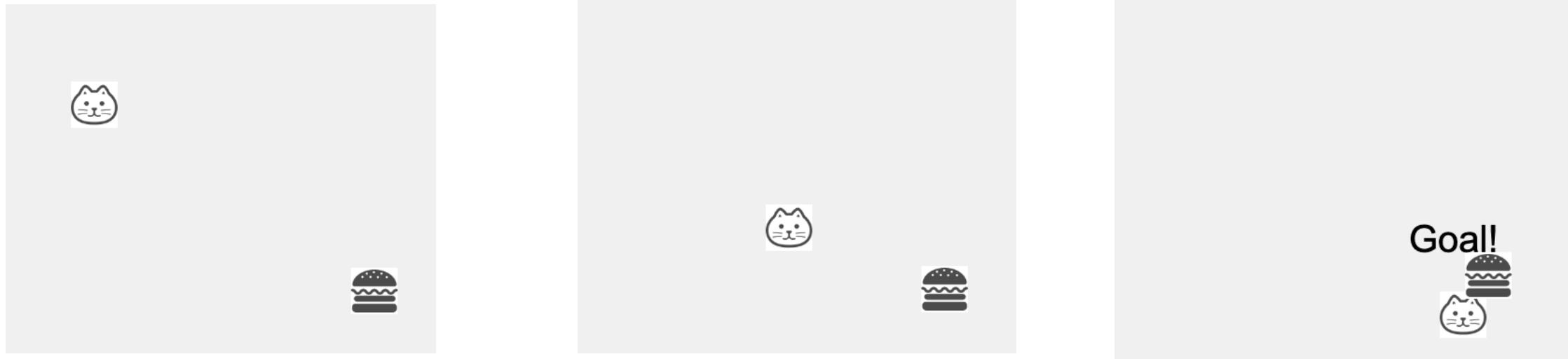
ラグ補償、ヒットボックス
垂直同期設定（VSync）
予測処理などを併用



サンプルゲーム

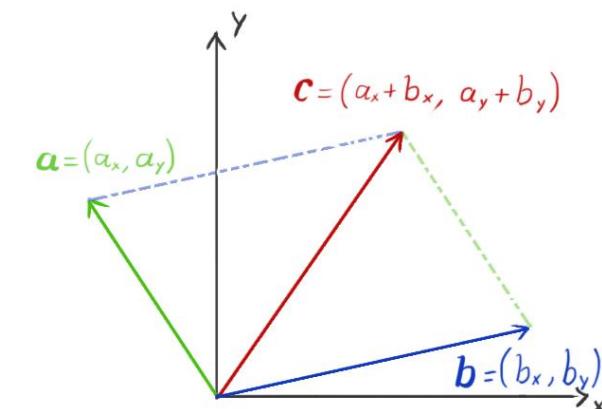
```
pythonsample > GameJS > js game.js > gameLoop
1  const canvas = document.getElementById('gameCanvas');
2  const ctx = canvas.getContext('2d');
3
4  const catImage = new Image();
5  catImage.src = 'cat.png'; // 猫の画像ファイルへのパス
6
7  const hamburgerImage = new Image();
8  hamburgerImage.src = 'hamburger.png'; // ハンバーガーの画像ファイルへのパス
9
10 let gameRunning = true;
11
12
13
14 const cat = {
15   x: 100,
16   y: 100,
17   width: 50,
18   height: 50,
19   speed: 1, // 移動速度を1に設定
20   dx: 0,
21   dy: 0
22 };
23
24 const hamburger = {
25   x: 400,
26   y: 300,
27   width: 50,
28   height: 50
29 };
30
31 function draw() {
32   ctx.clearRect(0, 0, canvas.width, canvas.height);
33   ctx.drawImage(catImage, cat.x, cat.y, cat.width, cat.height);
34   ctx.drawImage(hamburgerImage, hamburger.x, hamburger.y, hamburger.width, hamburger.height);
35 }
36
37 function update() {
38
39 }
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
```

```
pythonsample > GameJS > js game.js > ...
40   cat.y += cat.dy;
41
42   // キャンバスの境界を超えないように制限
43   cat.x = Math.max(0, Math.min(cat.x, canvas.width - cat.width));
44   cat.y = Math.max(0, Math.min(cat.y, canvas.height - cat.height));
45 }
46
47
48 catImage.onload = () => {
49   hamburgerImage.onload = () => {
50     requestAnimationFrame(gameLoop);
51   };
52 };
53
54 function checkCollision() {
55   const distance = Math.sqrt(Math.pow(cat.x - hamburger.x, 2) + Math.pow(cat.y - hamburger.y, 2));
56   if (distance < 50) { // ここでの距離は適宜調整してください
57     gameRunning = false;
58     ctx.font = '40px Arial';
59     ctx.fillStyle = 'black';
60     ctx.fillText('Goal!', canvas.width / 2 - 60, canvas.height / 2);
61   }
62 }
63
64 function gameLoop() {
65   if (gameRunning) {
66     update();
67     draw();
68     checkCollision();
69     requestAnimationFrame(gameLoop);
70   }
71 }
72
73 // キーボードイベントの処理 (keydown)
74 window.addEventListener('keydown', (e) => {
75   switch(e.key) {
76     case 'ArrowLeft':
77       cat.dx = -cat.speed;
78       break;
79     case 'ArrowRight':
80       cat.dx = cat.speed;
81       break;
82     case 'Space':
83       break;
84   }
85 }
```



- **猫とハンバーガーのオブジェクトの状態:**
 1. 位置 (x と y 座標) : これらはオブジェクトの画面上の位置を表します。
 2. 速度 (speed、 dx、 dy) : 猫のオブジェクトに関する速度と方向を表します。
 3. 画像 (catImage、 hamburgerImage) : これらはオブジェクトの視覚的表現を提供します。
- **ゲーム全体の実行On/Offの状態:**
 - **gameRunning 変数:** ゲームが進行中かどうかを表す布尔値です。
 - これはゲーム全体の「状態」を表しており、ゲームループの実行を制御します。

- 重力
 - オブジェクトが、地面へ落下する
 - ジャンプや落下の動きを再現
- 衝突
 - オブジェクトどうしが接触するときに発生
 - オブジェクトが互いを通過しないようにする
 - AABB、円形、多角形の衝突検出が一般的
- 応答と反発
 - 衝突後、オブジェクトは反発する
 - 弹力性や摩擦などの物理特性を計算



- 剛体ダイナミクス
 - 変形しない固体オブジェクトの動きを再現
 - 力、トルク、運動量などの物理法則を適用
 - 回転や移動をシミュレート
- ソフトボディダイナミクス
 - 変形可能なオブジェクト
 - 布やゲルのような柔らかい物質
 - 複雑な数学的計算が必要
- 流体ダイナミクス
 - 水や空気などの流体
 - 流れのパターン、圧力、乱流などを考慮
 - 複雑な数学的計算

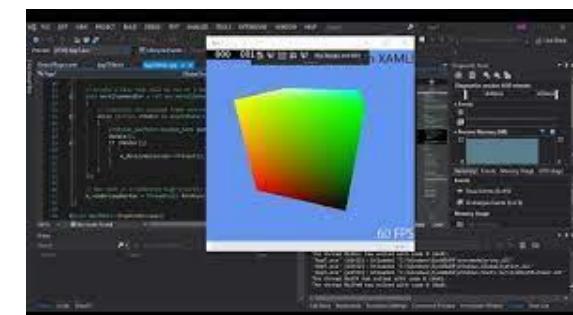
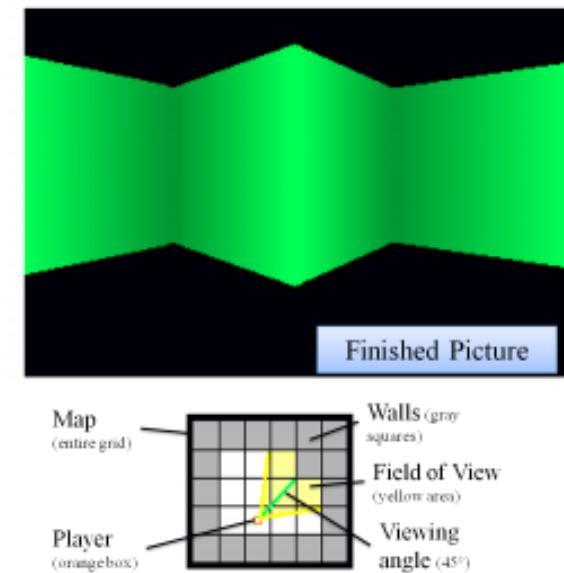
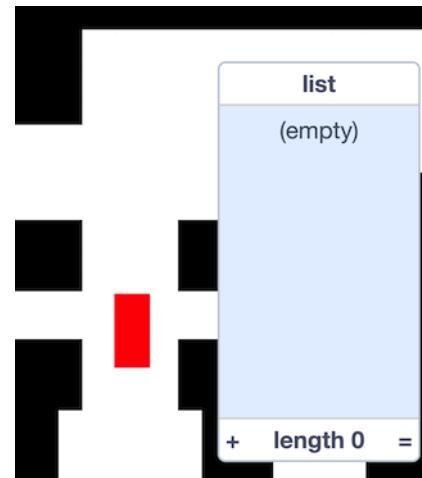
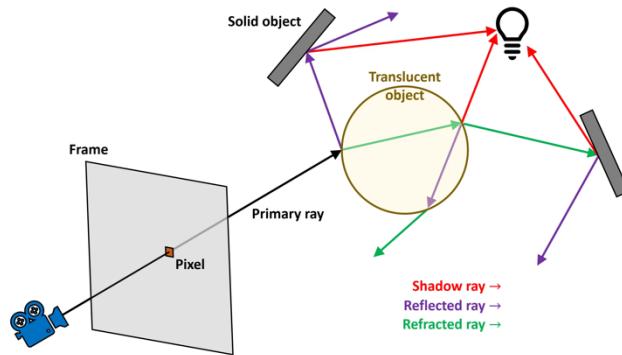


<https://www.youtube.com/watch?v=1o0Nuq71gl4>

サポートツールも存在する

記事：リアルタイムVFX用パーティクルベースのシミュレーション技術「**PhysX Flex**」の最新映像！剛体、ソフトボディ、流体もお手の物！
<https://developer.nvidia.com/content/new-flex-features>

- レイキャスト
 - オブジェクトから放出する線（レイ）
 - 視線や射撃軌道の計算、障害物検出に利用
- レイトレーシング
 - よりリアルな光の効果
 - 光の屈折や反射を追いかける



- 物理エンジンの利用
 - Box2D、UnityのPhysX、Havokなど
 - 複雑な物理計算を効率的に処理し、開発者の負担を軽減します
- 最適化とリアリズム
 - 計算コストの削減が必要
 - 目的に応じて、リアリズムを取る、特定の物理効果を省略
 - リアルな物理シミュレーションは計算コストが高いため、ゲームでは近似や単純化がしばしば行われます



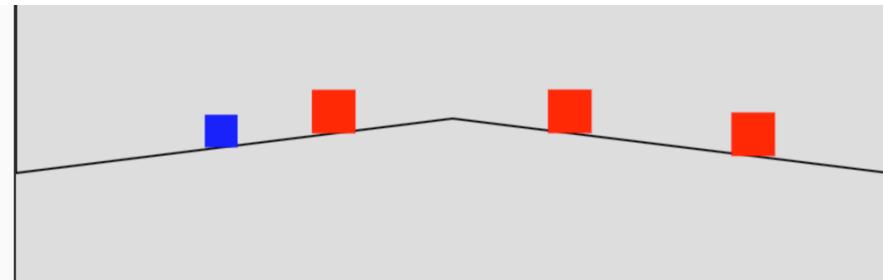
Microsoft Acquires Havok Physics from Intel

<https://www.techpowerup.com/216542/microsoft-acquires-havok-physics-from-intel>



https://www.youtube.com/watch?v=k7_1HQBnOtU

1-2時間で作れるゲームの例：コリジョン・物理演算



このゲームでは、簡易的な重力・反発などを実装しています。
下表のような概念を観察することで、2D物理を体感できます。

概念	説明	コードでの使用例	学べるポイント
重力 (GRAVITY)	物体を下方向へ加速させる力	<code>vy += GRAVITY;</code> 每フレーム下向きに速度を加算	重力加速度や自由落下の基本を理解し、垂直方向に加速する物理挙動を体感
摩擦・減衰 (FRICTION)	水平速度を少しずつ減らす（空気抵抗などのイメージ）	<code>vx *= FRICTION;</code> フレームごとに速度を少し減らす	動きがだんだん止まる挙動を表現し、現実的な減速を実感できる
反発係数 (BOUNCE_FACTOR, COLLISION_BOUNCE)	床や壁、他の物体と衝突した際の跳ね返り具合	<code>vy = -vy * BOUNCE_FACTOR;</code> 「1なら完全弾性」「0に近いほど減衰」など弹性衝突の仕組みを理解	「0に近いほど減衰」など弹性衝突の仕組みを理解
質量 (mass)	衝突時の運動量計算などに用いる物体の重さ	<code>player.mass = 1; obs.mass = 2;</code> 衝突演算で反映	運動量保存を簡単に導入し、質量が衝突結果にどう影響するか学習

1. 基本的な物理演算

重力と摩擦

```
// 物理定数
const GRAVITY = 0.3;           // 重力加速度
const FRICTION = 0.9;          // 摩擦係数

// 重力と摩擦の適用
player.vy += GRAVITY;         // 重力による落下
player.vx *= FRICTION;        // 水平方向の減速
```

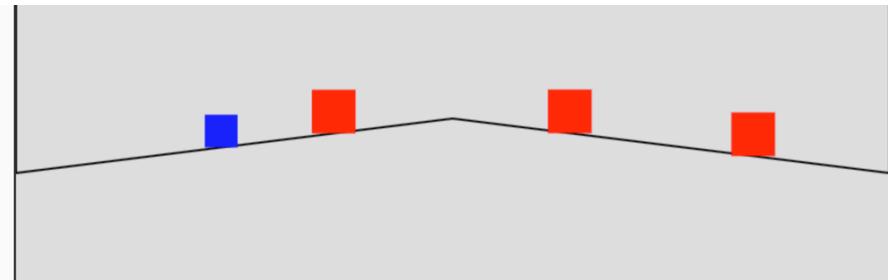
- 重力：キャラクターを下に引っ張る力
- 摩擦：水平方向の動きを徐々に減速させる

移動と速度

```
// 位置の更新
player.x += player.vx;        // x方向の移動
player.y += player.vy;        // y方向の移動
```

- 速度 (vx, vy) に基づいて位置を更新
- 毎フレーム この計算を繰り返す

1-2時間で作れるゲームの例：コリジョン・物理演算



このゲームでは、簡易的な重力・反発などを実装しています。
下表のような概念を観察することで、2D物理を体感できます。

概念	説明	コードでの使用例	学べるポイント
重力 (GRAVITY)	物体を下方向へ加速させる力	<code>vy += GRAVITY;</code> 每フレーム下向きに速度を加算	重力加速度や自由落下の基本を理解し、垂直方向に加速する物理挙動を体感
摩擦・減衰 (FRICTION)	水平速度を少しずつ減らす（空気抵抗などのイメージ）	<code>vx *= FRICTION;</code> フレームごとに速度を少し減らす	動きがだんだん止まる挙動を表現し、現実的な減速を実感できる
反発係数 (BOUNCE_FACTOR, COLLISION_BOUNCE)	床や壁、他の物体と衝突した際の跳ね返り具合	<code>vy = -vy * BOUNCE_FACTOR;</code> 「1なら完全弾性」「0に近いほど減衰」など弹性衝突の仕組みを理解	「1なら完全弾性」「0に近いほど減衰」など弹性衝突の仕組みを理解
質量 (mass)	衝突時の運動量計算などに用いる物体の重さ	<code>player.mass = 1; obs.mass = 2;</code> 衝突演算で反映	運動量保存を簡単に導入し、質量が衝突結果にどう影響するか学習

2. コリジョン（衝突判定）

基本的な矩形衝突

```
function isRectColliding(r1, r2) {  
    // 矩形同士が重なっているかをチェック  
    if(r1.x + r1.width < r2.x) return false;  
    if(r1.x > r2.x + r2.width) return false;  
    if(r1.y + r1.height < r2.y) return false;  
    if(r1.y > r2.y + r2.height) return false;  
    return true;  
}
```

- 二つの四角形が重なっているかを判定する最もシンプルな方法
- x軸とy軸それぞれで重なりをチェック

衝突後の処理

```
// 跳ね返り係数  
const BOUNCE_FACTOR = 0.3;  
  
// 衝突時の速度反転  
obj.vx = -obj.vx * BOUNCE_FACTOR;  
obj.vy = -obj.vy * BOUNCE_FACTOR;
```

- 衝突時に速度を反転させて跳ね返り表現
- BOUNCE_FACTORで跳ね返りの強さを調整

1-2時間で作れるゲームの例：アセット

- Github: <https://github.com/dmasubuchi/KITGames/tree/main/MazeGame10Maps>

アセットとは

ゲームを構成する素材（リソース）の総称：

例：画像（キャラクター、背景、アイテムなど）、
音声（BGM、効果音）、テキスト（シナリオ、設定
など）、マップデータ、設定ファイル

1. 開発効率の向上

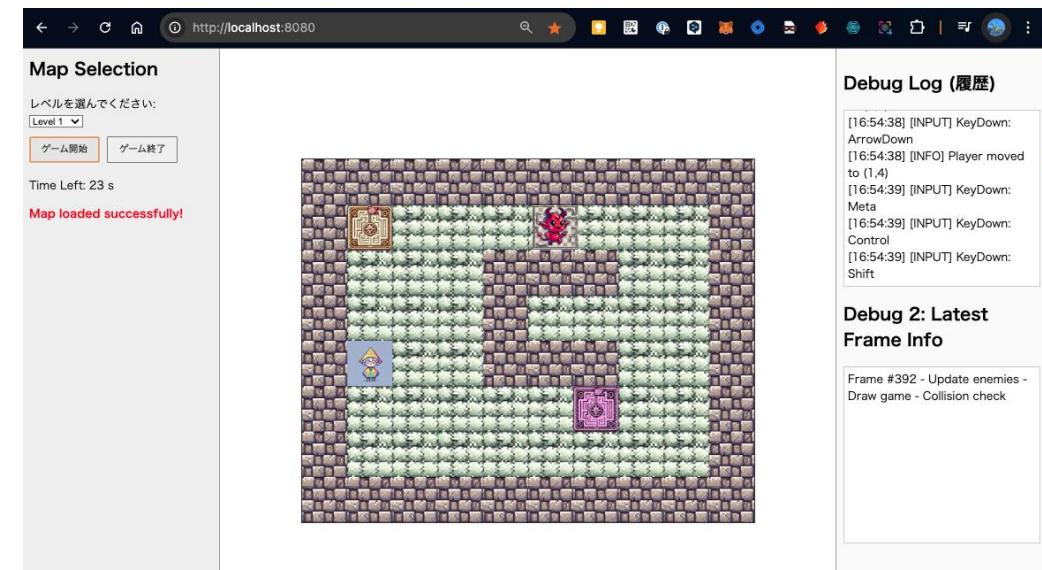
- 素材の差し替えが容易
- 複数の開発者での並行作業が可能
- アセットなしでも開発可能（フォールバック機能）

2. メンテナンス性

- アセットの変更が設定ファイルの編集だけで可能
- バージョン管理がしやすい
- 問題発生時の切り分けが容易

3. 拡張性

- 新規マップの追加が容易
- 新しい画像や音声の追加が簡単
- 異なるテーマへの対応が容易



1. ファイル構成

```
プロジェクト/
├── assets/          # アセットファイル
│   ├── player.png   # プレイヤー画像
│   ├── wall.png     # 壁の画像
│   ├── title.mp3    # タイトルBGM
│   └── game.mp3     # ゲーム中BGM
└── assets_config.json # アセット設定
└── MapLevel.json    # マップデータ
└── title.txt        # タイトル画面テキスト
```

- Chain-of-Thought アプローチによるアイデア展開
- 思考のプロセスを段階的に分解し、論理的にゲームコンセプトを構築する手法です。各ステップで得られた知見を次のステップに活かすことで、より実現可能性の高いアイデアを生み出すことができます。

以下の要素を持つ迷路探索ゲーム「KIT Explorer」のコンセプトを展開してください：

基本要件：

- ウェブブラウザで動作
- キーボードの矢印キーで操作
- シンプルな敵キャラクター
- 探索要素

各観点での詳細化：

1. ゲームの世界観
2. プレイヤーの目的
3. 核となる面白さ
4. 差別化ポイント
5. 教育的要素

次にアイディア検証、コンセプトマトリクス、ゲームニクスなどを作ります

KIT Explorerの基本メカニクスについて、以下の要素を詳細化してください：

1. プレイヤーの行動
 - 移動システム
 - 特殊アクション
 - 制限事項
2. 敵キャラクターの性質
 - . . .
3. 迷路の特徴
 - . . .
4. スコアシステム
 - 得点要素
 - クリア条件
 - ランキング実装

ユーザー体験のイメージを作ります

KIT Explorerのユーザー体験を以下の観点で設計してください：

1. チュートリアル設計

- 操作説明
- 初期ステージ

2. 難易度進行

- レベル構成
- 新要素の導入タイミング
- 達成感の演出

KIT Explorerのユーザー体験を以下の観点で設計してください：

3. UIデザイン

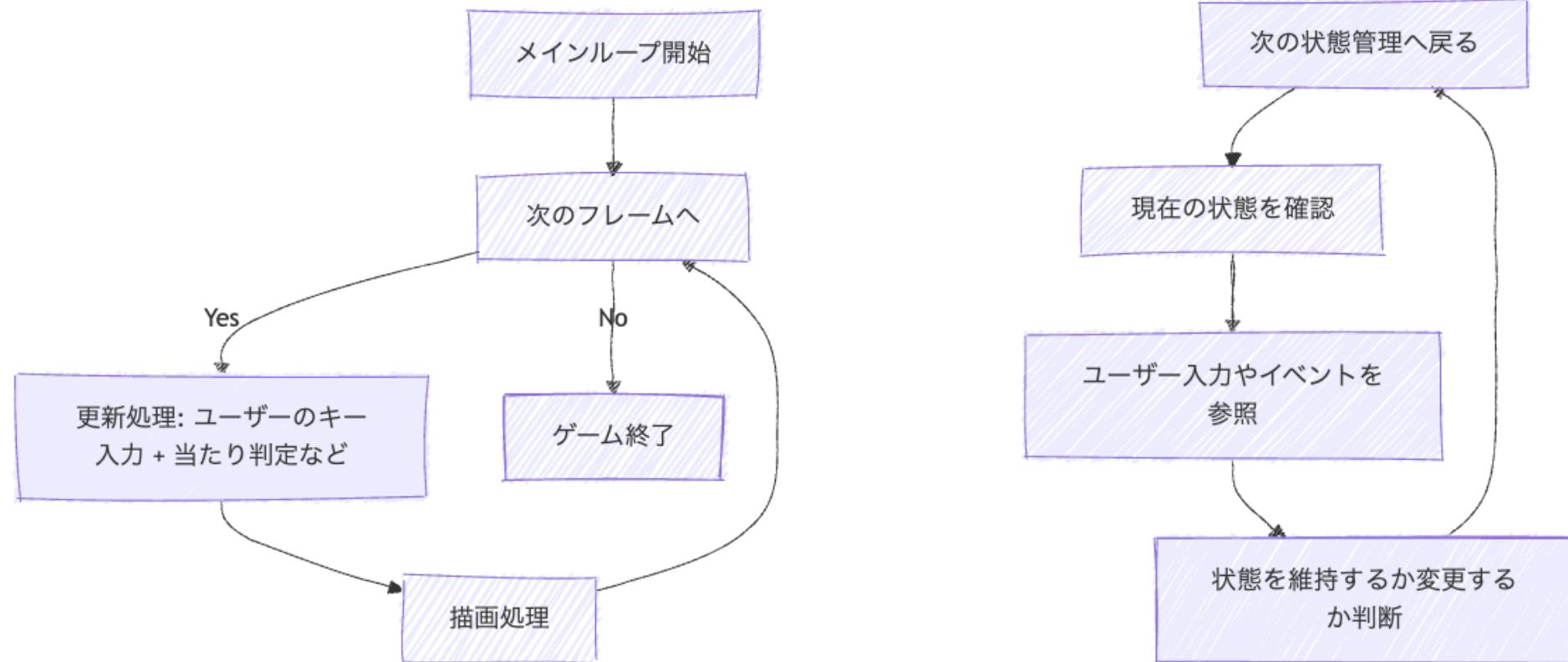
- 画面レイアウト
- 情報表示
- 操作フィードバック

4. サウンド要素

- BGM方針
- 効果音設計
- 演出タイミング

ゲームループなどの普遍的な基本の構造を設定する

必要な要素： 1. メインゲームループ 2. 状態管理 3. 更新処理 4. 描画処理



雑にAIに書かせても、基本設計があれば読みやすい



KurumeGameAI / MazeGame1 / script.js

KurumeGameAI / MazeGame1 / script.js

ポイント:

- ・**メインループ**: 每フレーム「更新→描画→次フレーム」の定型構造
 - ・**描画処理**: 背景→キャラ→UIなど、描画順を明示
 - ・**キーボード操作**: ユーザー入力を受けてキャラ座標やメニュー操作を制御
 - ・**関数分割**: `update()`, `render()`, `handleInput()`などに分けると可読性向上

個人ゲーム開発のポイント

誰が読んでも理解しやすい:
ゲーム開発の基本ロジックは世界共通

テンプレ化:
ループ/描画/入力の流れを守る

中身は拡張要素:

追加要素で味付けする
「音声, アニメ, 物理」などは後から

ロジックとコンテンツを分離

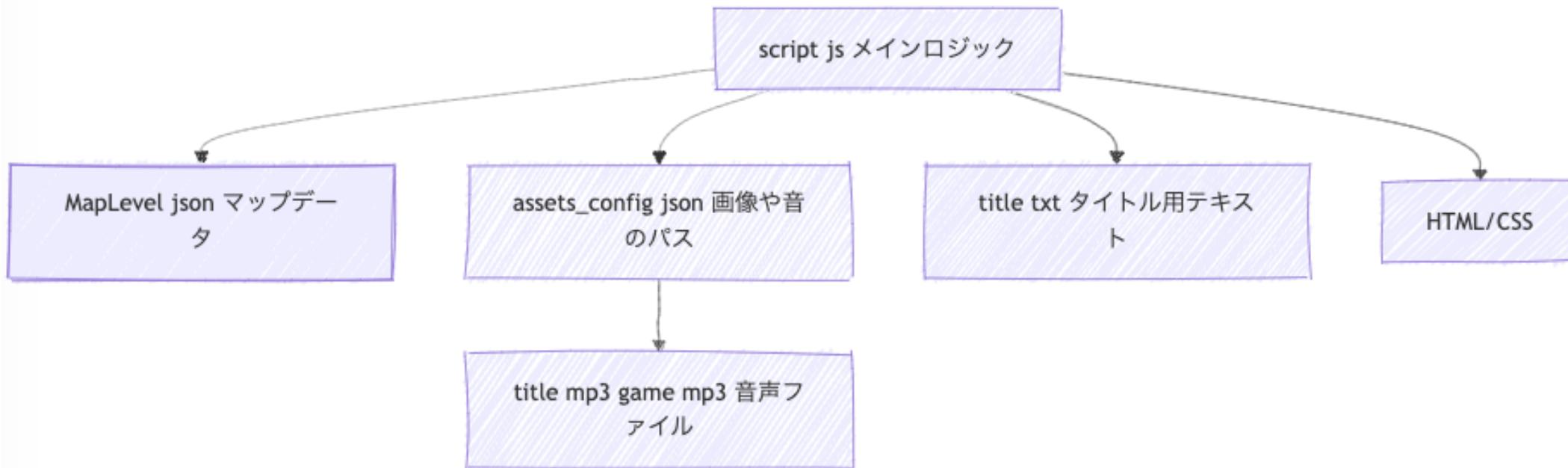
- ゲームの地図データやアセットリストを **JSON等の外部ファイル** に置き、コード側ではそのデータを読み取るだけにする。
- 修正や差し替えが容易になり、可読性も向上。

データ駆動設計を意識（サンプルではこれは未実装）

- キャラクターや敵のパラメータなどをハードコードせず外部定義
- AIに「JSONでデータを管理し、スクリプトはそれを参照する構造にして」と指示すればOK。

外部ファイルにする時はステップバイステップで行う

- はじめに「メインループ」「更新」「描画」を分けた小さなコードを作成
- 小さなコードの時は、外部ファイルの分離がしていない状態
- 段階的に「アセットを外部JSONに」「マップをJSONに」などの指示



https://mermaid.live/edit#pako:eNpVkbKA0EUhl9lmDrBwm4LwSSKRVllbtBht3Zi-zuLLNnvRBSOCNqJ0LETpFYCMElpDEo-DCHEHwLJ7tJkanO_PP95zJnQF3hcWpRPxanbsgkkF7DSYk5u3buyigDcpwT1M-oXIBPUb-h-kKtUX30Sb2-Qxp2h2VtfsJjA4rUoI_LZ_2A-hr1D6rf_ipfiTdtlucc8iNXpH4UVJ7F6Huub1Fd_j1N8eld9R2q2YatZUMEMSDwbSkLJu5QT1ZjF5RX6EyXc2Msvl0S8_eypNk2yRgSRWYCvPxJ-p7VNVIk406-ZBr9Peana7RqY1mnCZsMgzHzRYYg6FkCfc0ZYJPe6zlgaHOunQoKwA0T1PXWqBLhiNSIEE4fpSzb4D3opYIFICLZ_FuVG5F4GQnWoD5SJqNGPpoRBrZvgPFE2q4w

- ・ゲーム開発とAI の概要
- ・ゲームにおけるAI利用の変遷と拡張
- ・生成AIはゲーム開発に使えるの？
- ・ゲーム産業における象徴的なAI事例
- ・まとめと振り返り

プロシージャル生成（マップ／アイテム／コンテンツ）



大量のコンテンツを自動生成する仕組みの総称。制作コストと作業時間を大幅に削減し、リプレイ性を高める手法としてゲームで広く活用。

大まかな仕組み・構造、動きの例

1. 亂数生成

1. 亂数（ランダムシード）を受け取り、ゲーム開始ごとに異なる初期値を生成。
2. 例：Rogueではランダムに配置された部屋の位置や形状を組み立て、通路を結合してダンジョンを構築。

2. ルールベース

1. マップやオブジェクトの配置に関する一定の制約（例：部屋の最小／最大サイズ、通路の長さ、アイテムのレア度など）。
2. 乱数だけでなく、プレイヤー進行度や難易度バランスを考慮するケースもある。

3. ノイズ関数

1. Minecraftのように、地形形成にPerlin NoiseやSimplex Noiseを使い、自然風の連続性を再現。
2. 高さマップを生成し、丘や谷をランダムに配置。

代表的なアルゴリズム

- **Perlin Noise / Simplex Noise**
 - 自然な連続ノイズを生成するために用いられる。
- **Cellular Automata**
 - 2次元マップをセル単位でルール的に変化させ、洞窟や通路を生成。
- **Wave Function Collapse**
 - サンプルパターンを組み合わせてマップを埋めていく手法。タイル状のプロシージャル生成に向いている。

プロシージャル生成（マップ／アイテム／コンテンツ）

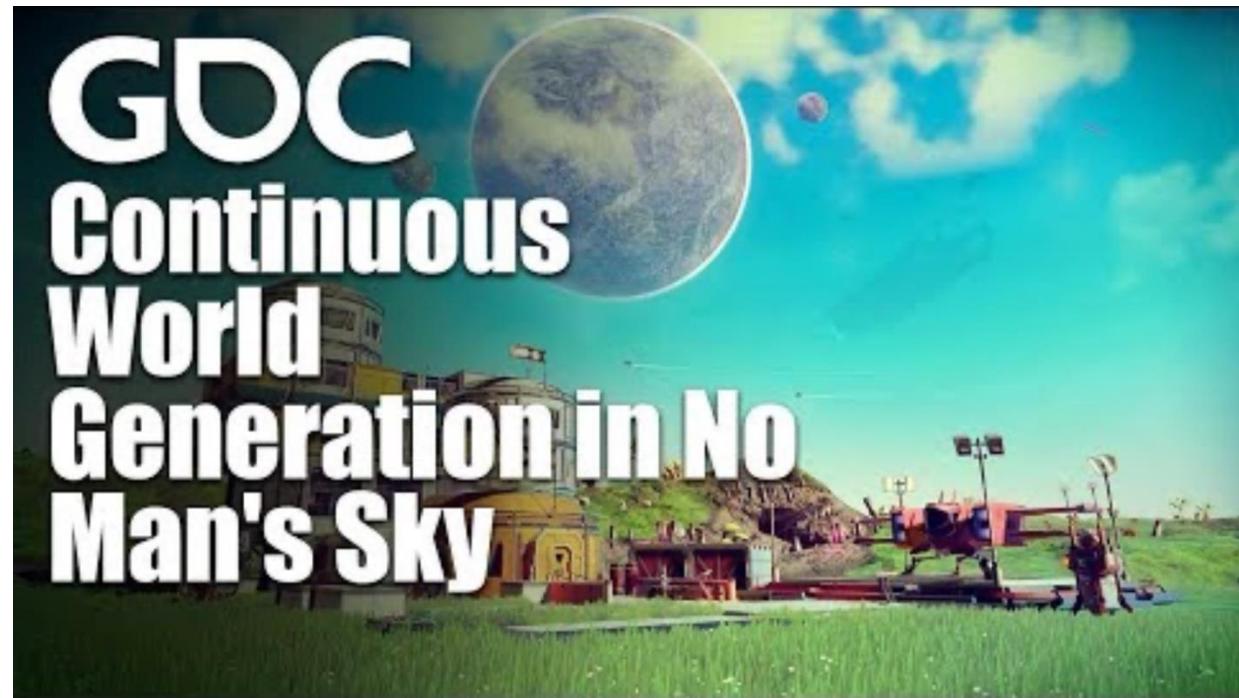


インディーでも導入が進み、手作業での世界構築コストを削減しつつコンテンツを拡張
建築・都市設計などゲーム以外の分野でも自動生成技術の応用が模索され盛り上がりを見せる

- **1980年 : Rogue**
 - ダンジョン生成で用いられた先駆的なプロシージャル技術。
 - 亂数 + ルールベースを組み合わせて毎回異なるマップを生成。
- **1997年 : Diablo**
 - ダンジョンやアイテムドロップの自動生成システムを本格導入。
 - ルールによる段階的なマップ生成とランダム要素でリプレイ性を確保。
- **2009年 : Minecraft**
 - 無限に近い地形を生成するアルゴリズムを実装。
 - ノイズ関数（Perlin Noise 等）を応用し、大規模な世界を構築。
- **2016年以降 : No Man's Sky**
 - 天体・地形・生態系・音楽まで自動生成する大規模システムを展開。
 - 多層にわたるルールや乱数を活用し、膨大な星系を構築。

No Man's Sky

- 開発背景:
 - Hello Games インディースタジオ 小規模チーム20人 開発初期4人 独自エンジン C++ PC/PS4同時開発
- 世界構造:
 - 球体惑星表面 立方体マッピング ボクセルベース 地形生成システム LOD管理 32x32x32メートル領域 オクツリー空間管理



No Man's Sky

- 生成パイプライン:
 - リアルタイム地形生成 マルチスレッド処理 並列実行 ボクセルデータ ポリゴン化 球体変換 物理メッシュ ナビメッシュ オブジェクト配置
- 技術的特徴:
 - デュアルコンタリング法 トライプラナー テクスチャリング コンピュートシェーダー メモリ最適化 パフォーマンス最適化 LODシステム



No Man's Sky

- アーティスト関連:
 - プロシージャル生成補完 アーティストコントロール ビジュアル評価 コンテンツ生成 アーティスティックビジョン
- 開発哲学:
 - 技術革新 アート・技術バランス 小規模チームの可能性 探索ゲーム開発 リアルタイム処理



Valveの『Left 4 Dead』シリーズに搭載されているAIディレクターは、プレイヤーの行動や状態をリアルタイムに監視し、敵の出現（ゾンビや特殊感染者）やアイテム配置、演出（BGM・SE）などを動的に制御するシステム



「回復アイテムが不足気味」 「プレイヤーがうまく連携してゾンビを排除している」などを判断し、スปーン数や配置アイテムを動的に調整。

Left 4 Dead 基本情報

- 2008年11月発売
- Valve Corporation開発
- 4人協力型ゾンビFPS
- PC/Xbox 360向け

GDC 2009での発表要点

1. AIディレクター

1. プレイヤーの状況に応じた動的調整
2. ドラマチックな展開の自動生成
3. リプレイ価値創出の仕組み

2. 技術革新

1. Source Engine活用
2. 動的な音楽/サウンドシステム
3. 協力プレイ最適化

ゲーム内テキストやイベントの分岐管理・自動生成を補助する技術。
大量のシナリオやクエストを統合的に管理することで、制作者の負担を軽減。

大まかな仕組み・構造、動きの例

1. 状態管理／フラグ管理

プレイヤーが獲得したアイテム、到達したシーンなどをトラッキングし、ストーリー進行を制御
イベント発生条件が整うと、シナリオ分岐を自動で振り分ける

2. 自然言語処理（LLM）

簡単なプロンプトや設定（世界観、キャラクターの関係性など）を入力すると、AIが下書きを生成
シナリオライターが補筆・修正して最終的に完成させる

代表的なアルゴリズム

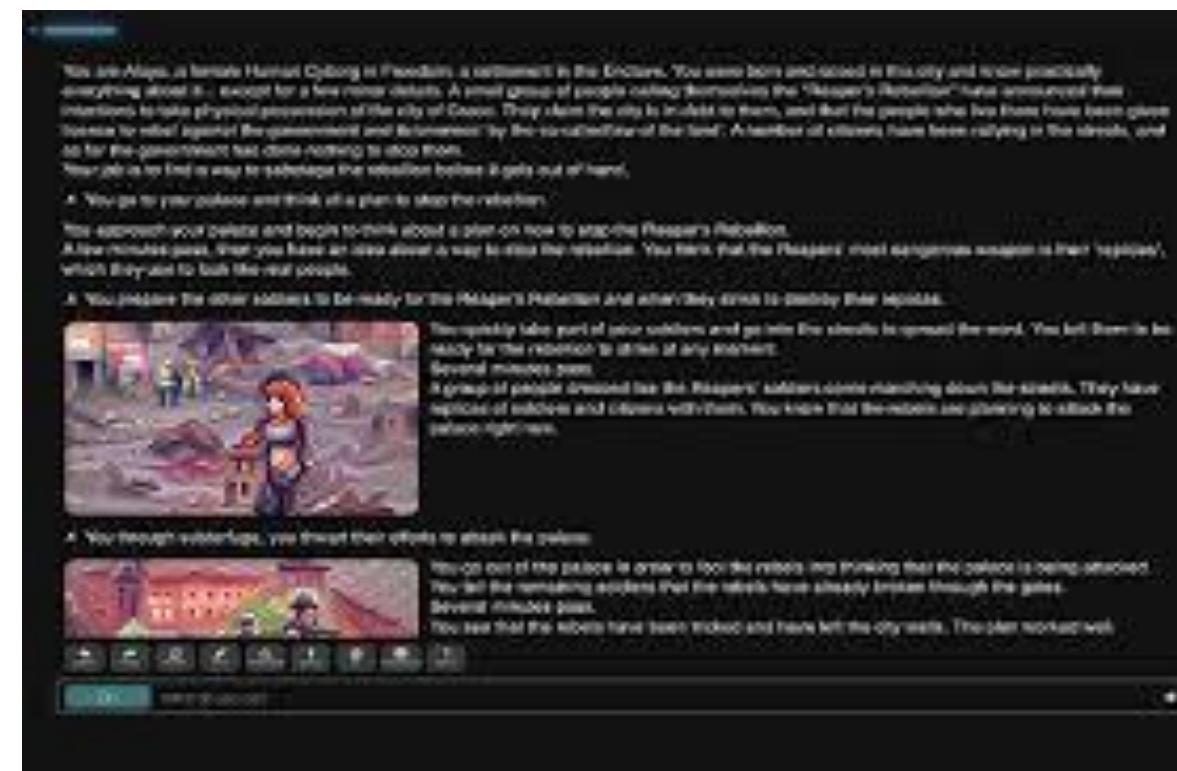
- ツリー／グラフ構造による分岐管理
 - マスターデータにイベント情報を記述し、ノードやエッジで分岐を表現。
- Markov Chain（マルコフ連鎖）
 - 単純な確率遷移を使い文章やストーリー展開を生成する際に使用。
- Transformerベース（例：GPT系）
 - 大量のテキストデータから学習し、高度な文章生成を行う。ゲームシナリオにも応用可能。

企画段階から商業タイトルまで、プロトタイプ生成にAIを活用する流れが一般化、LLMは、かなり広く使われてます

- **2002年：Neverwinter Nights**
 - 「モジュール」形式でユーザーが独自シナリオを構築可能。
 - 公式ツールにおいてイベントの自動配置支援が導入。
- **2014年：ファイナルファンタジーXIV（拡張運用段階）**
 - 大量クエストをスクリプト管理する際、シナリオ分岐やイベント配置をツール化。
 - プレイヤーの進行状況に応じたクエスト分岐を自動管理。
- **2022年以降：大規模言語モデル（LLM）の活用**
 - ストーリーテリング支援や会話フロー原案の自動生成。
 - AIアシスタントがクエスト文書やNPCのバックストーリーを下書き、人間が仕上げるフローが増加。

AIダンジョン

- AI Dungeonの革新点：
 - オープンエンドなストーリー生成
 - プレイヤーの選択による物語分岐
 - テキストベースの無限の可能性
 - 動的なワールド構築



NPC（非敵キャラクター）対話・演出



自然言語処理(NLP)を駆使し、ゲーム内でNPCがプレイヤーと会話するインタラクションを演出。
テキストの大量生成やNPCの個性付けにより、ゲーム世界の没入感を高める。

大まかな仕組み・構造、動きの例

- **音声合成・音声認識**
 - ユーザーが音声入力した内容をテキストに変換し、AIが解析。
 - NPCが応答する際には音声合成エンジンでリアルタイムにボイスを生成。
- **会話履歴・コンテキスト管理**
 - 直近の会話だけでなく、長期的な履歴（プレイヤーの選択やNPCとの関係性）を保持し、発話内容を変化。
- **LLM (ChatGPTなど) の推論**
 - 内部でNPCの設定や世界観をパラメータ化し、会話内容を動的に生成。

代表的なアルゴリズム

- **音声認識エンジン（例：CMU Sphinx, Kaldi, DeepSpeech など）**
 - プレイヤーが喋った内容をテキスト化。
- **RNN / Transformer ベースの言語モデル**
 - NPCのテキスト応答を生成する中核部分。
- **Dialogue Manager（対話管理システム）**
 - 会話の状態と遷移を制御し、シームレスな対話を実現。

ChatGPTのようなモデルの登場で進化が加速

ユーザーとの自由な会話演出を目指す試作が多数発表されており、業界全体で注目度が上昇

- **2011年：Siri（ゲームではなく、スマホ領域だが技術的転用に注目があつまる）**
 - 音声対話技術が注目を集め、ゲーム向けの音声入力・対話演出への研究が進むきっかけに。
- **2017年：Ubisoftの対話アシスタントプロトタイプ**
 - NPC同士の会話やプレイヤーとの対話を自然言語処理で生成する試み。
 - 大量のテキストを自動生成して開発チームの手間を削減。
- **2023年：ChatGPTなどのLLMベースの試作**
 - NPCの応答やクエスト説明をリアルタイムに生成するデモが各社で進行。
 - 会話履歴に基づいてNPCの個性を維持するシステムなどを検討中。

- NPCルーチンシステムの特徴： リアルタイム動的シミュレーション 24時間のアクティビティサイクル 環境に応じた行動パターン適用 都市の生活感を演出

技術的な側面： 行動パターン
データベース NPCインスタンス
管理 リアルタイム経路探索 環境
相互作用の処理

AIの革新ポイント： 大規模な人
口シミュレーション コンテキス
トに応じた行動選択 動的なスケ
ジューリング 環境認識と対応



https://www.youtube.com/watch?v=u_5WIltkG8

- AI工コシステム：動物行動シミュレーション 生態系の食物連鎖再現 天候や時間による行動変化 種間の相互作用

NPCシステム：詳細な日常行動パターン 環境に応じた会話生成感情表現システム 記憶による行動変化

環境相互作用：ダイナミックな天候影響 地形による行動制限 季節変化への対応 環境破壊と再生



Red Dead Redemption 2 Reimagined with Real Life AI

https://www.youtube.com/watch?v=pOBj3Ps_ybQ

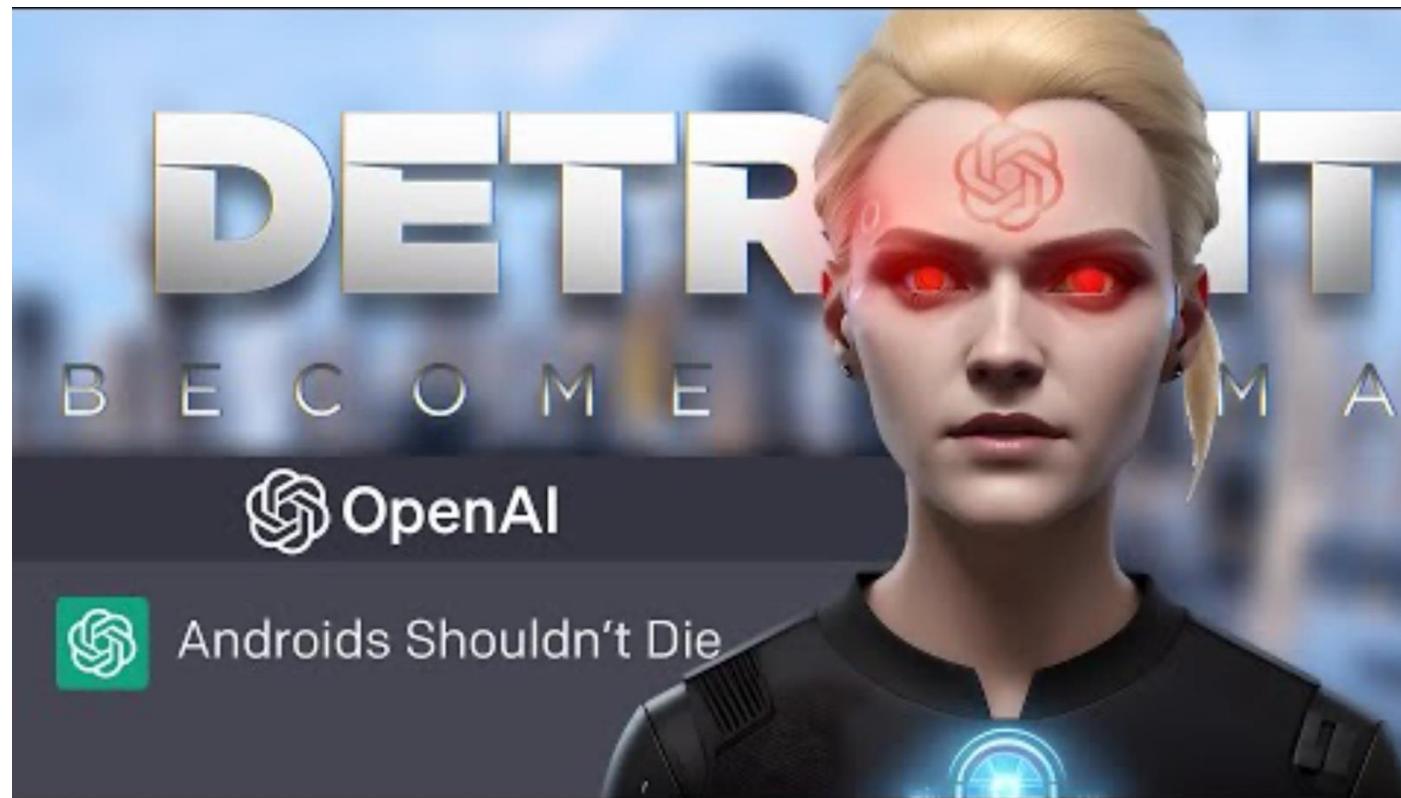
- Inworld AIの革新点：
 - 動的な会話生成システム
 - NPCに個性と記憶を保持
 - プレイヤーの文脈理解
 - リアルタイムの会話フロー



Mass AI character generator powered by LLMs | Inworld AI Unity Demo

<https://www.youtube.com/watch?v=Aev65jeXozY>

ゲーム「Detroit: Become Human」をプレイする様子を記録したYoutube動画
ただし、プレイヤー自身ではなく、ChatGPTにゲームの選択を委ねている。
AIによる、AIゲームの攻略として話題に。AIに邪悪な振る舞いを設定するが予想外の展開に・・・



アニメーション・物理演出

キャラクターの動きや物理挙動を自動化・効率化し、リアリティとバリエーションを高める技術群。従来は職人的にモーションを制作していた部分を、物理シミュレーションや機械学習が補完する。

大まかな仕組み・構造、動きの例

- **物理演算+アニメーション制御**
 - Euphoriaの場合、キャラクターの骨格に物理エンジンが作用し、衝突や転倒などをリアルタイム生成。
 - 従来のラグドールに制御アルゴリズムを加えることで、“ぶざま”にならない動きに仕上げる。
- **モーションマッチング**
 - 過去のモーションキャプチャデータから“今の状況”に最も合う姿勢や動きを検索し、連続的に適用。
 - フットIK（足裏位置補正）などと組み合わせることで、フィールドに適した動きを生成。
- **自動リギングツール**
 - モデルに骨格を自動で設定し、各頂点に対するウェイトを推定。
 - アニメーターの初期作業を大幅に削減。

代表的なアルゴリズム

- **Inverse Kinematics (IK)**
 - キャラクターの手足を目標地点に合わせて自動的に関節角度を算出。
- **ラグドール物理**
 - 骨格を剛体やジョイントとして扱い、衝突や重力などを再現。
- **検索ベースのモーションマッチング**
 - フィーチャーベクトルを用いて“最適なポーズ”を検索するアルゴリズム。

リアルタイム物理演算やモーションマッチングが映画VFXにも展開され、マルチメディア業界で需要増。ゲーム内のモーション品質向上と開発効率アップに寄与し、事例が増えている。

- **2008年 : Euphoria (NaturalMotion)**
 - ラグドール物理とアニメーションを組み合わせたキャラクター挙動システム。
 - 手作業モーションを減らし、動きのバリエーションを自然に演出。
- **2018年 : モーションマッチング (EA Sportsなど)**
 - 大量のモーションキャプチャデータから、AIが適切な動きを自動選択して繋ぎ合わせる。
 - モーションの遷移を滑らかにし、開発コストを削減。
- **2022年 : AI支援リギングツール**
 - 三次元モデルに自動で骨格やウェイトを設定するソフトウェアが普及。
 - UnityやUnreal Engineのプラグインで、AIによる自動アニメーション補完機能が登場。

ゲームエンジンのリアルタイム性と、機械学習によるコード解析・テキスト生成を統合し、制作全体を支援する技術。
ゲーム外の映像分野やツール開発にも大きな影響を与えている。

大まかな仕組み・構造、動きの例

- **リアルタイムモーションキャプチャ**
 - カメラやセンサーから得たデータを即座にゲームエンジンへ反映し、仮想環境でプレビュー可能。
 - バーチャルセット上でキャラクターの動きや表情をリアルタイムに確認しながら撮影／制作が進行。
- **AIによるコードアシスタント**
 - コミット時にソースコードを解析し、パターンマッチングや機械学習で潜在的バグを検出。
 - 自動修正案を提示し、開発者が承認する形で品質向上・開発効率化。
- **AIテキスト生成 (Ghostwriterなど)**
 - NPCのセリフやフレーバーテキストを自動作成。
 - 翻訳や多言語対応を含めたドキュメントの初期草案を迅速に用意。

代表的なアルゴリズム

- **機械学習ベースのコード解析**
 - 大量のリポジトリを学習したモデルが“よくあるバグパターン”を高精度で検知。
- **リアルタイムレンダリングエンジン (例 : Unreal Engine, Unity)**
 - レンダリング最適化と入力デバイス統合により、インタラクティブな制作環境を構築。
- **NLP (自然言語処理)**
 - NPCやイベント台詞を生成し、翻訳や文体変換にも対応。

コード解析やテキスト生成AIが開発工程のボトルネックを大幅に緩和
バグ検出から翻訳まで幅広い領域をカバーし、数多くのスタジオが導入検討中

- **2015年 : Epic GamesのLive Link**
 - モーションキャプチャやフェイシャルキャプチャを取り込み、エンジン上に反映。
 - ゲームエンジンを映画撮影やPV制作にも応用する事例を生む。
- **2019年 : Ubisoft La Forge 「Commit Assistant」**
 - コードのバグや潜在的問題をAIが自動検出し、修正を提案。
 - 大規模コードベースでの人的負担を削減。
- **2022年 : Ubisoft 「Ghostwriter」**
 - NPCのセリフやイベント台詞をAIが原案生成し、ライターが校正・編集。
 - 多言語展開や大量テキスト制作を効率化。

ゲームの品質保証(QA)やデバッグ工程をAIエージェントが代行・支援する技術。
大規模ゲームにおけるテスト工数の削減と品質向上を両立。

大まかな仕組み・構造、動きの例

1. AIエージェントによる模擬プレイ

1. 仮想プレイヤーが実機・エミュレーター上でボタン操作や移動を自動実行。
2. 多彩なパターンの入力を試し、バグ発生状況や難易度調整の不備を検出。

2. 強化学習を用いた攻略探索

1. 学習済みのAIがゲームのゴールやスコアを最大化するために試行錯誤。
2. 通常のプレイヤーが行わない操作や極端なプレイスタイルも網羅しやすい。

3. サーバー負荷検証

1. 大量のAIエージェントを同時に接続して、通信遅延やサーバーの耐久性をチェック。

代表的なアルゴリズム

• ルールベースのプレイシミュレーション

- 事前に定義されたテストシナリオを順番に実行し、結果を収集。

• 強化学習 (Q-learning, PPOなど)

- AIエージェントが学習ベースでゲームをプレイし、未知のバグや問題を発見。

• 大規模並列シミュレーション

- クラウドリソースを利用し、同時に多数のプレイを実行して膨大なデータを収集・分析。

AAAタイトルの膨大なテスト工数やサーバー負荷テストにAIが貢献する事例がでてきました
初期バージョンから自動化を組み込む流れが定着し、大手を中心に導入が加速している

- **2010年代中盤：自動プレイテスト**
 - スクリプトやAIエージェントがゲームを模擬プレイし、バグや難易度の偏りを検出。
 - AAAタイトルのQAを効率化。
- **2018年：EAの自動テストプラットフォーム**
 - 強化学習ベースでゲームのデバッグを部分的に自動化。
 - マルチプレイ要素もAIが模擬プレイヤーとしてテストし、サーバー負荷を検証。
- **2021年：UnityのGame Simulation**
 - クラウド上で大量の自動プレイセッションを回し、データを可視化。
 - AIがバグ発生箇所や詰まりやすいポイントをレポート。

Game Engine (Unity, Unreal Engine) のAI対応



UnityのAIシステムは、ゲームエンジン内に設計された人工知能コンポーネントとシステムの総称です。このシステムの主な目的は、NPCに環境認識能力、意思決定能力、そして複雑な動作を実行する能力を付与することにあります。

Learn more about our AI products

- UNITE/2024** Unlock new gameplay using Runtime AI with Unity Sentis
- UNITE/2024** Improving your in-Editor workflows with Unity Muse
- UNITE/2024** Say hello to new 2D workflows and AI enhancements

Unlock new gameplay using Runtime AI with Unity Sentis

Discover how AI models can transform your game and bring previously impossible features to life on user devices.

WATCH NOW →

Improve your in-Editor workflows with Unity Muse

Learn how you can access an AI-powered helper and a range of generative AI capabilities for prototyping, all within the Unity Editor.

WATCH NOW →

Say hello to new 2D workflows and AI enhancements

Watch this video for practical tips on eliminating repetitive tasks and simplifying supporting props thanks to Unity's new AI-powered 2D workflows.

WATCH NOW →

Unreal Engineは、包括的なAIツールセットを提供しています。

AIの行動を定義するビヘイビアツリー、AIの記憶を管理するブラックボードシステム、移動可能範囲を定義するNavMesh、AIに視覚や聴覚を与える認識システムです。

Join our community, grow your knowledge and learn from others!

Sign in

Don't have an Epic Games account? [Sign up](#)

DEV COMMUNITY

- Unreal Engine
- Forums
- Documentation
- Learning
- Snippets

Unreal Engine 5.5 Documentation

Filter by title

Artificial Intelligence

Describes the systems available within Unreal Engine that can be used to create believable AI entities in your projects.

ON THIS PAGE

- General Topics
- Machine Learning

General Topics

- Behavior Trees
- Documents the Behavior Trees asset in Unreal Engine and how it can be used to create Artificial Intelligence (AI) for non-player characters in your projects.
- MassEntity
- MassEntity is a gameplay-focused framework for data-oriented calculations.
- Navigation System

- ゲーム開発とAI の概要
- ゲームにおけるAI利用の変遷と拡張
- 生成AIはゲーム開発に使えるの？
- ゲーム産業における象徴的なAI事例
- まとめと振り返り

AIはゲーム開発に革新をもたらす一方、目的に応じた適切な活用方法の選択が重要になってきています。

技術的側面

- GPU開発とAIの相乗的発展
- ゲームエンジンのAI機能の進化
- リアルタイム処理技術の革新
- 機械学習基盤としてのゲーム環境

開発プロセス的側面

- プロシージャル生成の活用
- テスト工程の自動化
- NPCの高度な行動制御
- 大規模なコンテンツ生成

教育・学習側面

- 生成AIによる開発の敷居低下
- プログラミング学習との相乗効果
- AIツールを活用した試作の容易さ
- 個人開発からの段階的な成長機会

ビジネス・産業側面

- 商用タイトルでのAI利用
- 生成系AIはリスクも多い
- ゲームの仕組みが複雑になる
(個人開発とは対照的)

- ゲームは単なる娯楽ではなく、AIアルゴリズムの開発と検証のための理想的な実験場となり得ます。
- MicrosoftのProject MalmoやGoogleのDeepMindが示すように、ゲーム環境でのAI研究は、実世界の問題解決への重要なステップとなります。
- ゲームは現実世界とは異なり、
以下↓の利点を持つ実験環境

明確な境界と制約

- 物理法則の定義が明確
- 状態の変化が予測可能
- 環境の範囲が限定的

制御された複雑さ

- 必要な要素だけを実装可能
- 段階的な複雑化が可能
- データの収集が容易

興味を持った分野に少し関わってみてください
きっと面白い発見と新しい可能性が待っています

技術的なアプローチの選択肢として 
問題解決のヒントとして アイデアの発想源として 
研究テーマのインスピレーションとして 

ゲーム開発とAI

株式会社リンクトブレイン
技術戦略顧問
増渕大輔

