**開発プロジェクト計画：シャドウバース対戦支援AI**

**ステップ0：開発環境の構築 (Sprint 0)**

まず、開発を始めるための土台を整えます。

1. **バージョン管理システムの導入:**
   * **Git** をインストールし、**GitHub** にてプライベートリポジトリを作成します。これにより、コードの変更履歴を管理し、安全に開発を進められます。
2. **プログラミング言語・ライブラリのインストール:**
   * **Python 3.10** 以降をインストールします。
   * コマンドプロンプトまたはターミナルで、以下のコマンドを実行し、必要なライブラリをインストールします。

Bash

# 仮想環境の作成を推奨 (venv)

python -m venv shadowverse-ai-env

source shadowverse-ai-env/bin/activate # Mac/Linux

# shadowverse-ai-env\Scripts\activate # Windows

# ライブラリのインストール

pip install opencv-python numpy

pip install pytesseract

pip install Pillow

pip install pyqt6

pip install torch torchvision torchaudio

* + **Tesseract OCR本体**を別途PCにインストールし、Pythonから呼び出せるようにPATHを通しておきます。

1. **テスト用データの収集:**
   * 様々なクラス、様々な状況（序盤、中盤、終盤）のシャドウバースのスクリーンショットを**最低でも50枚以上**収集します。これが後の画像認識モジュールの開発とテストに不可欠です。

**開発ロードマップ（全12スプリント案）**

**フェーズ1：画像認識モジュールの確立 (Sprint 1〜3)**

**ゴール：ゲーム画面を正確に読み取り、データ化する**

* **Sprint 1: 画面キャプチャと領域特定**
  + [ ] ゲームウィンドウを自動で探し、その領域のみスクリーンショットを撮る機能を実装。
  + [ ] OpenCVのテンプレートマッチングを使い、手札、PP、リーダー情報など、主要UI要素の**座標を特定**するロジックを組む。
  + **成果物:** 指定したUI要素を切り出して保存できるスクリプト。
* **Sprint 2: カード情報のOCR読み取り**
  + [ ] 切り出したカード画像から、Tesseract OCRで**カード名とコスト、攻撃力、体力**を読み取る。
  + [ ] OCRの認識精度を上げるため、画像の二値化、ノイズ除去などの前処理を試行錯誤する。
  + **成果物:** 1枚のカード画像から、その情報をテキストとして出力する関数。
* **Sprint 3: ゲーム状態の完全データ化**
  + [ ] これまでの機能を統合し、1枚のスクリーンショットから盤面全体の情報を**JSON形式で出力**する。
  + [ ] 収集したテスト用データセットを使い、認識精度を評価し、改善する。
  + **成果物:** screenshot\_to\_json() という関数を持つ、安定した画像認識モジュール。

**フェーズ2：プロトタイプの構築 (Sprint 4〜5)**

**ゴール：AIの思考結果を画面に表示する、動くプロトタイプを完成させる**

* **Sprint 4: 基本UIとルールベースAIの実装**
  + [ ] PyQtを使い、クリックしても下のゲームに影響しない**半透明のオーバーレイウィンドウ**を作成する。
  + [ ] 非常にシンプルなAI（**ルールベースAI**）を実装する。（例：「出せるカードの中で一番コストが高いものを出す」「攻撃可能なフォロワーで相手リーダーを攻撃する」など）
  + **成果物:** 画像認識の結果をルールベースAIが判断し、推奨手をコンソールに出力するプログラム。
* **Sprint 5: オーバーレイ表示の実現**
  + [ ] ルールベースAIの判断結果をオーバーレイウィンドウに描画する。
  + [ ] 例：推奨カードの座標に**緑色の枠線**を表示する、攻撃対象へ**矢印**を描画する。
  + **成果物:** **バージョン0.1** となる、実際にプレイを支援（？）してくれるアプリケーション。

**フェーズ3：AIの知能向上 (Sprint 6〜12)**

**ゴール：機械学習モデルを導入し、AIを「賢く」する**

* **Sprint 6-7: 学習データの収集と準備**
  + [ ] トッププレイヤーの対戦動画やリプレイサイトから、学習データ（「この盤面」で「この行動」が正解）を収集する。
  + [ ] 収集したデータを (盤面状態, 行動) のペア形式に変換・整形するスクリプトを作成する。
* **Sprint 8-10: モデルの学習と評価**
  + [ ] PyTorchを使い、盤面の良し悪しを判断する「**価値ネットワーク**」と、次善手を予測する「**方策ネットワーク**」を設計・実装する。
  + [ ] 準備したデータセットでモデルを学習させる。学習の進捗を監視し、ハイパーパラメータを調整する。
* **Sprint 11-12: AIモデルの統合と最適化**
  + [ ] 学習済みモデルをAI推論エンジンに組み込み、ルールベースAIと置き換える。
  + [ ] リアルタイムでの推論速度を計測し、遅い場合は**ONNX Runtime**の導入を検討して高速化を図る。
  + **成果物:** **バージョン1.0** となる、機械学習ベースの高度な対戦支援AI。

**注意すべきリスクと対策**

* **ゲームのアップデート:** UIが変更されると画像認識が機能しなくなります。座標のハードコーディングを避け、設定ファイルで変更可能にするなど、メンテナンスしやすい設計を心がけてください。
* **法的・倫理的リスク:** Cygamesの利用規約を定期的に確認し、決してゲームの自動操作やメモリ改ざんといった禁止行為に繋がらないよう、設計原則を厳守してください。

さあ、まずは**ステップ0：開発環境の構築**から始めましょう。準備が整い次第、Sprint 1のタスクに着手してください。健闘を祈ります！