ETロボコン2018

デベロッパー部門 アドバンストクラス

地区：東京　　地域：神奈川県川崎市

チームNo. 331　　チーム名：FCT

所属：株式会社富士通コンピュータテクノロジーズ

チーム紹介、目標、意気込み

私たちは株式会社富士通コンピュータテクノロジーズの有志社員3名で構成されたチームです。当社は富士通グループの組込みシステム開発・サービスを提供する事業を担っています。

メンバーは1年目の社員2名と5年目の社員1名から構成されています。3名とも過去にETロボコンの参加経験があり、これまでのモデリング・プログラミング経験、ハードウェアの知識をフル活用して大会に臨みます。ETロボコンへのやる気と情熱は、どのチームにも負けません。チーム名のFCTは富士通コンピュータテクノロジーズの頭文字からとりました。

地区大会ではモデルに多くの課題を残し、競技は難所の完全攻略を逃しました。全国大会では、モデル・ソフトウェアに磨きをかけた成果を発揮し、優勝を目指します。

モデルの概要

- Rコース攻略を達成するためにブロック並べゲームの攻略を中心に記載したモデル図。
- 開発目標をRコースリザルトタイム-11.8秒に設定し、実現するために必要な機能をまとめる。
- ゲーム分析では、運搬経路探索に幅優先探索を利用し、探索された経路の中からより運搬リスクの低い経路が選択される。
- 走行体のシステムはシナリオをもとに動作するように設計されており、走行タイムエリア、ブロック並べエリア、直角駐車の攻略は、各シナリオをシナリオ実行する共通のモジュールに入力することで実現できる。
- 制御戦略はゲーム内での走行体の走行、AIアンサーの攻略について検討した結果を中心に記述。
- 要素技術では、画像解析に挑戦した結果をまとめる。画像解析により走行体はPCから送信されるブロックの色と位置の情報を使用することでブロック並べを攻略することができる。

モデルの構成

モデリング対象をRコースとして記述。

1.要求分析

・開発目標を「Rコースリザルトタイム-11.8秒」に設定し、目標達成のための大まかな機能要件をユースケース図を用いて記述。

・さらに、機能要求・非機能要求について要求図を用いて分析結果を記載。

2.分析モデル

・3ステップに分けてゲーム要素を分析した結果とそこから導き出された課題を記述。

・要素定義から指針を検討し、ゲーム攻略手順を記載。

・幅優先探索を用いたブロック並べエリアの攻略手順をシーケンス図とゲーム要素のオブジェクト図で表現。

3.分析モデル/設計モデル

・2の分析モデルに引き続き、指針から導いたゲームの解法を記述。

・システムの設計方針を示し、パッケージ図を用いて大まかにソフトウェア構成を表現。

4. 設計モデル

・システムの詳細設計をクラス図で表現。

・構造モデルの振る舞いをシーケンス図、ステートマシン図、アクティビティ図を用いて記述。

5.制御モデル

・要求モデル、分析モデルで記述した内容を実現するための制御戦略、要素技術として以下の項目を記述。

-ブロック並べエリア内の走行体の制御技術
-AIアンサー攻略技術
-PCと走行体の通信
-HSVを用いた色判別
-画像を用いたカラーブロックの色判別

1. 要求モデル(p.1)

FCT

1.1 開発目標

Rコースリザルトタイム-11.8秒

※Lコースはモデリング対象外のため省略

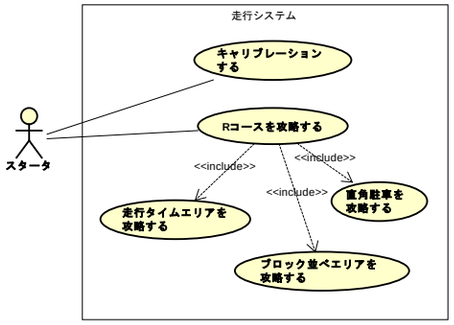
各地区大会の結果から全国大会で優勝するためには、走行タイムリアを16.2秒で走行し、全難所をクリアすることが必要となる。したがって、以下の通り各エリアの目標クリアタイムとボーナスタイムを設定した。

- 走行タイムエリア：16.2秒
  全国ランキングより、走行タイムの速い上位3チームの平均タイムは17.2秒。チャンピオンシップ大会では出場チームがこの平均タイムより1秒速く走行したと仮定。
- ブロック並べ：85秒
  ブロック有効移動4個、パワースポット設置5個
  ボーナスタイム23秒
- 直角駐車：15秒
  ボーナスタイム5秒

リザルトタイム：16.2 - 23 - 5 = -11.8秒

1.2 機能要件

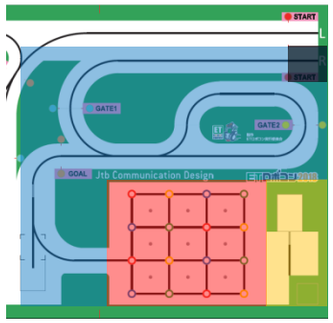
Rコースリザルトタイム-11.8秒を実現するために必要なユースケースを以下のように抽出した。例として、Rコースを攻略するユースケース記述を示す。その他のユースケース記述は紙面の都合上省略する。



ユースケース名	Rコースを攻略する
概要	Rコースを攻略する
アクター	スタータ
事前条件	キャリブレーションが終了している
事後条件	走行システムが停止している
基本フロー	1. スタータはタッチセンサを押してスタート指示を出す 2. システムは走行タイムエリアを攻略する 3. システムはブロック並べを攻略する 4. システムは直角駐車を攻略する

エリアと要素技術

下図のように黒色部分をスタートエリア、青色部分を走行タイムエリア、赤色部分をブロック並べエリア、黄色部分を直角駐車エリアと定義する。要求図で抽出した要素技術と各エリアの関係を表を用いて整理した。



要素技術 ＼ エリア	スタートエリア	走行タイムエリア	ブロック並べエリア	直角駐車エリア
カラーキャリブレーション	○	—	—	—
アームモータ制御	○	○	○	○
明度PID	—	○	○	○
バンド補正	○	○	○	○
車輪制御最適化	—	○	○	○
距離計	—	○	○	○
方位計	—	—	○	○
パターン走行	—	—	○	—
HSV	—	○	○	○
カメラキャリブレーション	○	—	—	—
経路探索	—	—	○	—
画像解析	—	—	○	—

1.3 要求分析

「1. 開発目標」と「2. 機能要件」を踏まえ、要求図を用いて機能要求や機能要求に付随する要求を抽出した。

凡例

目標
システムを開発するにあたって、達成を目指す水準

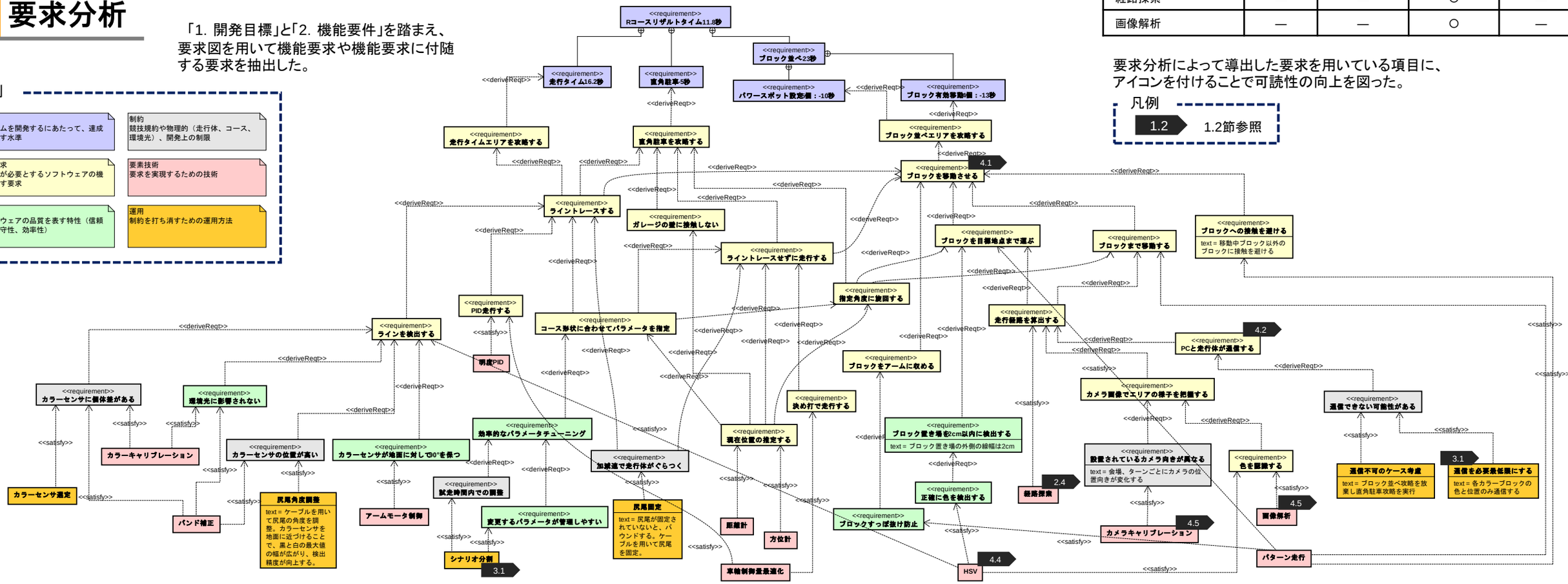
制約
競技規約や物理的（走行体、コース、環境光）、開発上の制限

機能要求
ユーザが必要とするソフトウェアの機能を表す要求

要素技術
要求を実現するための技術

品質
ソフトウェアの品質を表す特性（信頼性、保守性、効率性）

運用
制約を打ち消すための運用方法



凡例

1.2 1.2節参照

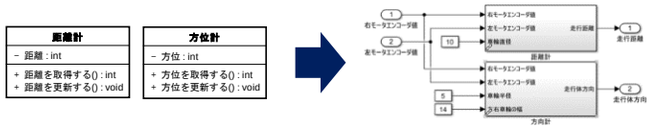
要求分析によって導出した要求を用いている項目に、アイコンを付けることで可読性の向上を図った。

1.4 モデルベース開発

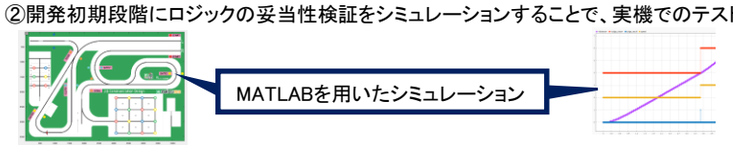
従来の開発では、モノができてから動作検証を行っていたため、実機動作に問題が発生した場合、手戻りや修正に時間を要していた。

そこで今回のETロボコンでは、組み込みシステム開発における新しい開発モデルであるモデルベース開発を採用して設計～評価を行う。モデルベース開発では、以下のようなメリットが得られた。

①設計シートで記載した1つのクラス図を、ほぼそのまま1つのブロック図と対応させているため、機能追加や変更などの保守性が向上した。



②開発初期段階にロジックの妥当性検証をシミュレーションすることで、実機でのテスト工数を大幅に削減した。



③モデル化した設計書をそのまま自動コード生成することで実装の工数削減・品質向上。
④上流工程で品質を高めているため、下流工程の手戻りなどが削減可能。

2. 分析モデル(p.2)

FCT

2.1 ゲームの要素定義

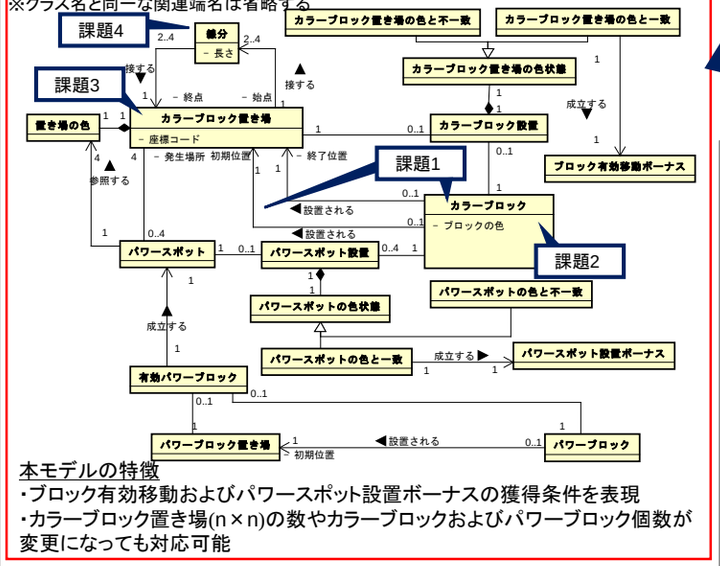
Step1～Step3の手順でブロック並べゲームの構成要素の特徴や関係を整理する。

Step1 ブロック並べゲームを分析する上で必要な要素を整理する。

①カラーブロック置き場同士は線分を介して接続
②初期位置コードによりパワーブロックおよびカラーブロックの置き場の初期位置が決定
③有効パワーブロックはパワーブロックがパワーブロック置き場に置かれていると成立
④パワースポットは有効パワーブロックを囲む四角形のカラーブロック置き場に成立
⑤ブロック有効移動ボーナスはカラーブロックの色とカラーブロック置き場の色が一致すると獲得
⑥パワースポット設置ボーナスはカラーブロックの色とパワースポットの色(=カラーブロック置き場の色)が一致すると獲得
⑦パワースポットは1つのカラーブロック置き場に重複して発生

Step3 Step2の内容から本ゲーム要素についてクラス図で整理する。
図中の課題の詳細説明は「2.3指針」で記述する。

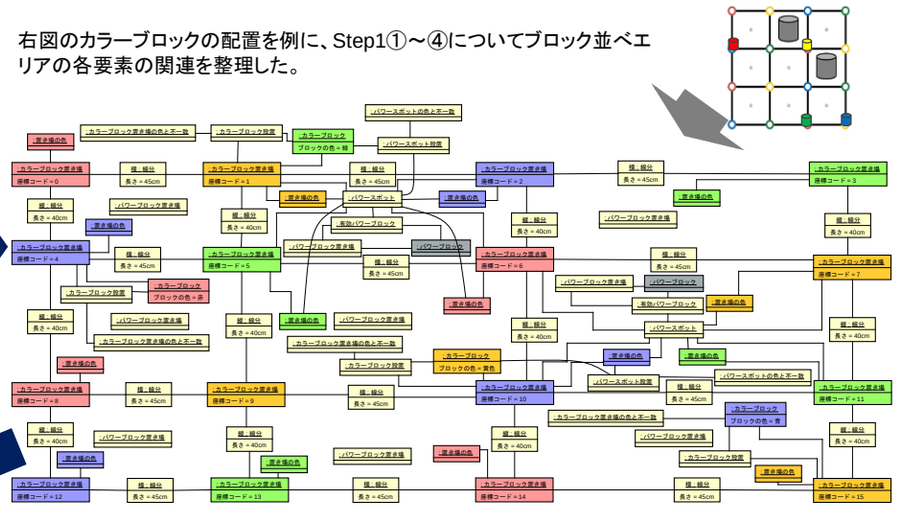
※クラス名と同一な関連端名は省略する



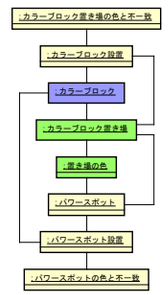
本モデルの特徴
・ブロック有効移動およびパワースポット設置ボーナスの獲得条件を表現
・カラーブロック置き場(n×n)の数やカラーブロックおよびパワーブロック個数が変更になっても対応可能

Step2 Step1の内容について、オブジェクト図を作成して整理する。

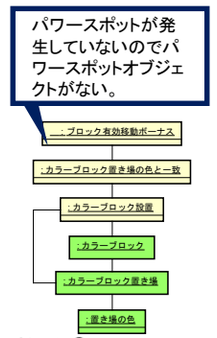
右図のカラーブロックの配置を例に、Step1①～④についてブロック並べエリアの各要素の関連を整理した。



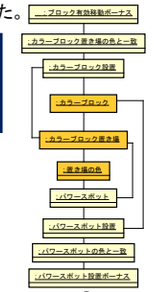
次に1つのカラーブロックに対するボーナス獲得パターンとその時の要素間の関連を整理した。カラーブロック置き場の色とカラーブロックの色、パワースポットの色とカラーブロックの色それぞれ2つの観点で色の一致状況を分析することで、Step1の⑤～⑦について表現できた。



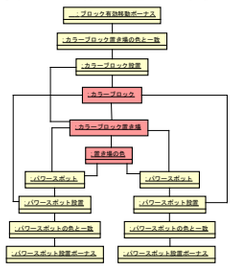
パターン①
パワースポットに色が異なるカラーブロックを設置した例



パターン②
パワースポットでないカラーブロック置き場に色の一致しているカラーブロックを設置した例



パターン③
同色かつパワースポットであるカラーブロック置き場にカラーブロックを設置した例



パターン④
同色かつパワースポットが重複しているカラーブロック置き場にカラーブロックを設置した例

ブロック有効移動獲得数	パワースポット設置獲得数		
	0個	1個	2個以上
0個	パターン①	発生しない	発生しない
1個	パターン②	パターン③	パターン④

2.2 走行体の動作定義

基本方針:
一定の動作パターンを作り、その組み合わせでエリアを走行させる

パターン1:直進

パターン2:右折

パターン3:左折

走行体はパターン1～3の動作パターンのみでエリア内を走行可能。

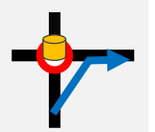
問題点
・ブロック置き場にブロックが置かれている場合、通過不可
・ブロックが置かれていない経路を選択すると移動距離が増える可能性大
・ブロックの取りこぼし、走行途中で落とすことを防ぎたい
・競技時間は最大2分

ブロックを回避するような動作パターンが必要
走行体のブロック保持、不保持を考慮する必要

動作パターンを追加定義

パターン4:直進

パターン5:右折

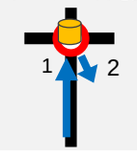


パターン6:左折

パターン7:180度旋回

パターン8:180度旋回

パターン9:ブロックを置く



走行体と動作パターンの関係

	動作パターン
ブロックを取る	1、2、3、7
ブロックをよける	4、5、6、8
ブロックを置く	9
ブロックを持っている	1～8
ブロックを持っていない	1～9

2.3 指針

「2.1ゲームの要素定義」「2.2走行体の動作定義」を活用してブロック並べの攻略手順を検討した。

課題1
走行前にブロックの色は公開されない。走行開始後にブロックの色を認識する必要がある。
指針1 4.5
エリア内に設置されているカメラシステムを用いてブロックの色・位置情報を認識する。

課題2
4つのカラーブロックを運搬する順序を決定する必要がある。
指針2
パワースポットの数が多く、走行体の位置から最も近いカラーブロックを選択する。

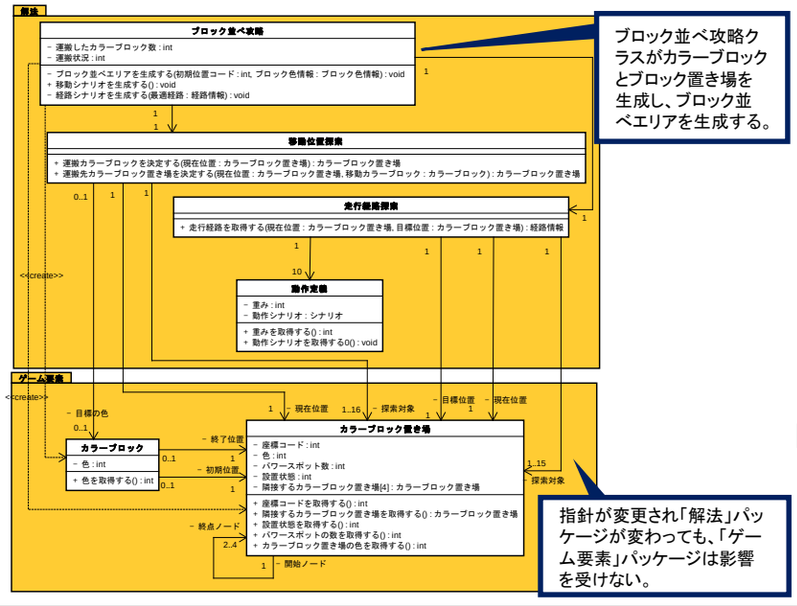
課題3
4つのブロック置き場から1つ選択する必要がある。
指針3
パワースポット設置ボーナスの獲得を目標として、カラーブロック置き場を選択する。

課題4
どのような経路でカラーブロックをカラーブロック置き場に移動させるか。
指針4
走行体の動作定義で示したパターンを組み合わせて、もっとも移動コストが低い経路を選択する。

全てのカラーブロックを運搬する

2.4 解法

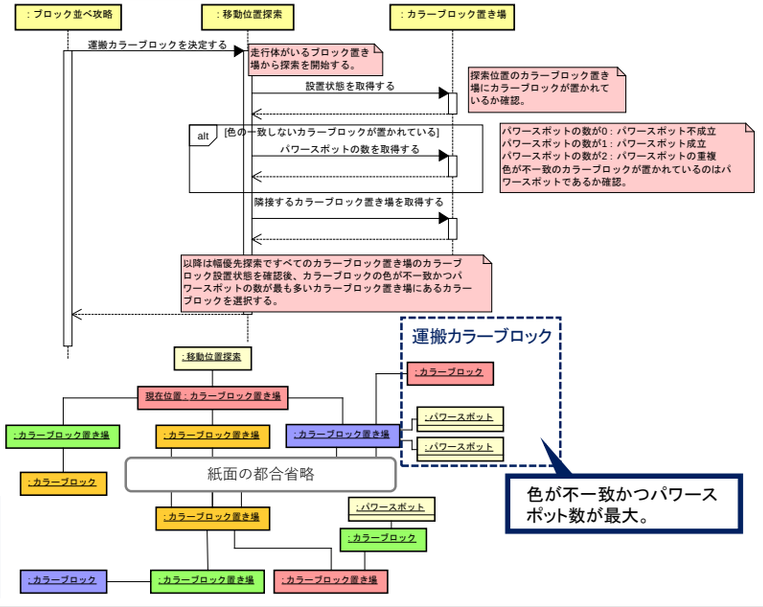
「2.1 ゲームの要素定義」から不要なクラスなどを整理して「ゲーム要素」パッケージとした。また、「2.3 指針」を実現する要素は「解法」パッケージとしてまとめ、 指針に沿った解法を検討した。ブロックを運搬する経路探索に幅優先探索を利用する。指針2を運搬カラーブロックを決定するシーケンス図、指針3を運搬先カラーブロック置き場の決定するシーケンス図、指針4を最適経路を探索するシーケンス図で表現した。



ブロック並べ攻略クラスがカラーブロックとブロック置き場を生成し、ブロック並べエリアを生成する。

指針が変更され「解法」パッケージが変わっても、「ゲーム要素」パッケージは影響を受けない。

運搬カラーブロックを決定する

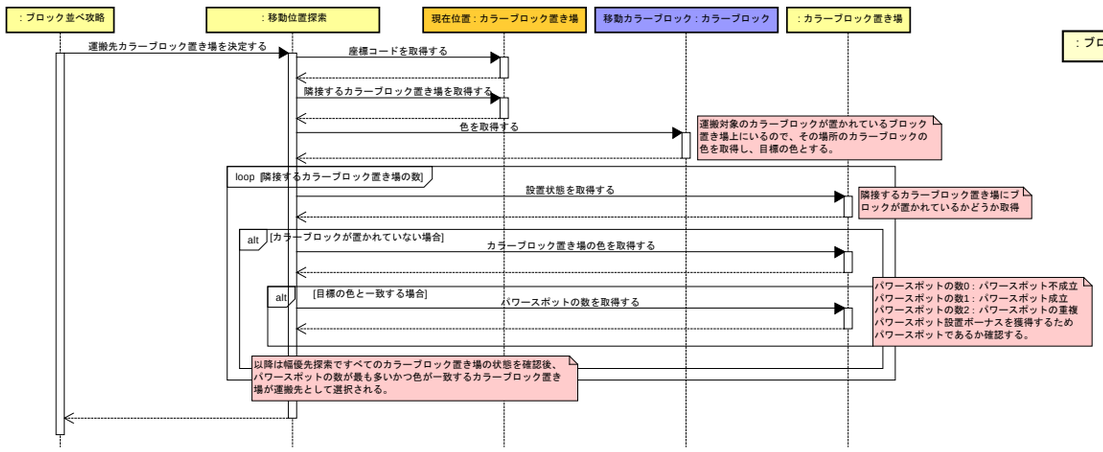


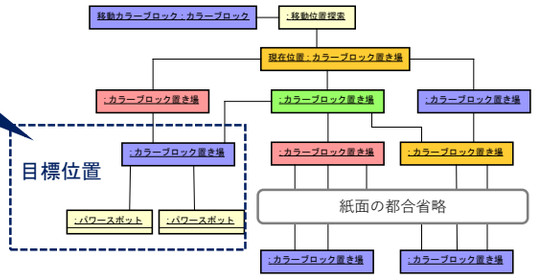
色が不一致かつパワースポット数が最大。

3. 分析/設計モデル(p.3)

2.4 解法(p.2のつづき)

運搬先カラーブロック置き場を決定する

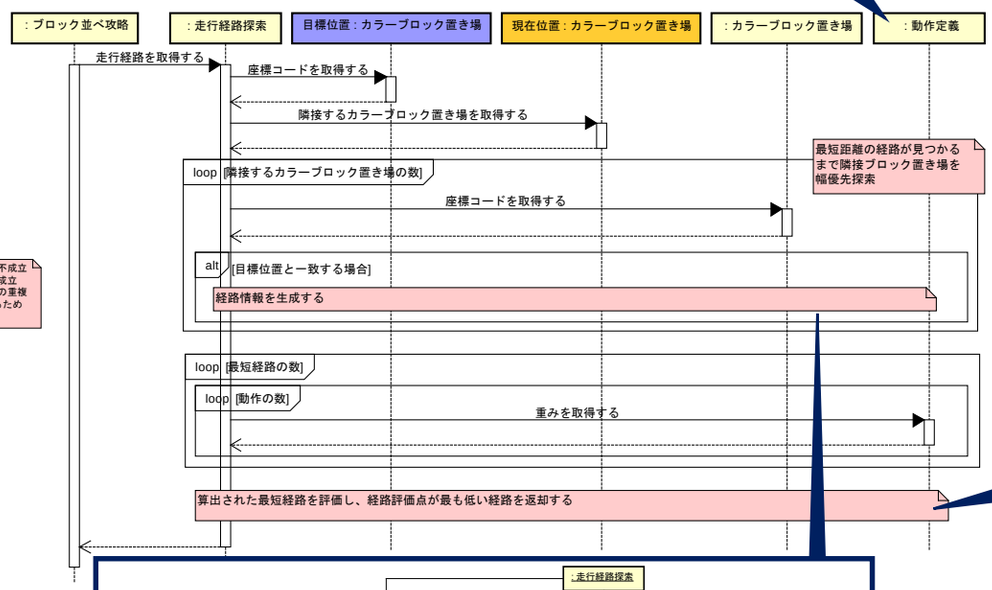




移動カラーブロックと同色かつパワースポットが重複していてカラーブロックが置かれていないブロック置き場

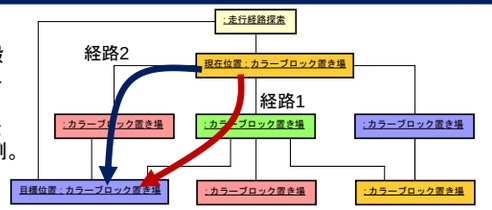
目標位置

最適経路を探索する



算出された最短経路を評価するため、「2.2動作定義」で定義した動作パターンに重みを付与する。
重み0:パターン1、4
重み1:パターン2、3、5、6
重み2:パターン7、8

幅優先探索を用いて、運搬する距離が最小となるすべての経路が探索される。右図は最短距離が2である経路1と経路2が算出された例。

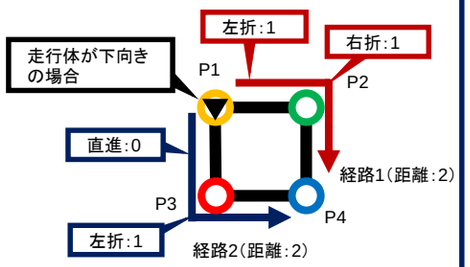


経路評価
探索されたすべての経路は走行体の動作定義で定義した重みを利用して、評価される(評価方法は下式のとおり)。その中で最もコストの低い経路が最適経路として選択される。

$$C = d + w_1 + w_2 + \cdots + w_{n-1} + w_n$$

C:コスト　d:距離　w:重み

【運搬経路の評価例】
経路1(P1→P2→P4)と経路2(P1→P3→P4)が探索された場合、経路の評価例を以下に示す。



経路1　2+1+1=4
経路2　2+0+1=3
よって、この場合は探索された経路の中から経路2が最適経路として選択される。

3.1 設計方針

シナリオ設計
走行体の動作は、状況に応じて振る舞いをさせる必要がある。振る舞いを動作とその終了判定に分類して、それらを1つのシナリオとして扱う。走行体は連続したシナリオを実行することでRコースを攻略する。シナリオで使用する走行制御・終了判定のパラメータに以下の表に示す。

動作制御用パラメータ	説明
動作制御種別	決め打ち走行 or PID走行を指定
速度	モータへの出力量を指定
旋回量	左右モータへの補正量を指定
Kp、Ki、Kd値	PID走行に必要な係数を指定
エッジ	ラインの右 or 左、目標値(黒白灰組み合わせ)を指定

終了判定用パラメータ	説明
終了判定種別	色 or 距離 or 方位を指定
終了判定値	判定で使用する目標値を指定
終了判定条件	判定で使用する関係演算子(等しい or 以上 or 以下)を指定

下図は連続して動作する場合のシナリオ例
①左旋回しながら5cm前に進む。
②黒色まで直進する。
③その場で180°左旋回する。
④赤色までラインに沿って前に進む。
⑤右旋回しながら黒色まで前に進む。
⑥10cm後ろに進む。

シナリオ①:シナリオ	シナリオ②:シナリオ	シナリオ③:シナリオ	シナリオ④:シナリオ	シナリオ⑤:シナリオ	シナリオ⑥:シナリオ
走行制御種別 = 決め打ち走行 速度 = 30 旋回量 = 10 Kp値 = - Ki値 = - Kd値 = - エッジ = - 終了判定種別 = 距離判定 終了判定値 = 5cm 終了判定条件 = 判定値以上	走行制御種別 = 決め打ち走行 速度 = 30 旋回量 = 0 Kp値 = - Ki値 = - Kd値 = - エッジ = - 終了判定種別 = 色判定 終了判定値 = 黒色 終了判定条件 = -	走行制御種別 = 決め打ち走行 速度 = 0 旋回量 = 30 Kp値 = - Ki値 = - Kd値 = - エッジ = - 終了判定種別 = 方位判定 終了判定値 = 180° 終了判定条件 = 判定値以上	走行制御種別 = PID走行 速度 = 40 旋回量 = - Kp値 = 0.8 Ki値 = 0.01 Kd値 = 0.07 エッジ = 右エッジ(黒白) 終了判定種別 = 色判定 終了判定値 = 赤色 終了判定条件 = -	走行制御種別 = 決め打ち走行 速度 = 10 旋回量 = -10 Kp値 = - Ki値 = - Kd値 = - エッジ = - 終了判定種別 = 色判定 終了判定値 = 黒色 終了判定条件 = -	走行制御種別 = 決め打ち走行 速度 = -30 旋回量 = 0 Kp値 = - Ki値 = - Kd値 = - エッジ = - 終了判定種別 = 距離判定 終了判定値 = -10cm 終了判定条件 = 判定値以下

Bluetooth通信設計 4.2
①何の情報を走行体に送信するか？
PC側でブロック並べ攻略のシナリオを生成して送信する場合、データ量や通信回数が多くなり、データロスする可能性がある。また、通信中にBluetooth通信が切断されると走行不能になる。よって、PCと走行体の通信は必要最低限とし、PCはブロックの色・位置のみを送信する。

②いつ走行体とBluetooth通信するか？
スタート直前までブロックの色は確定していない。また、走行エリアタイムエリアを走行中にBluetooth通信すると、走行体の動作が不安定になる。以上よりスタート後の停止した状態で通信を行う。

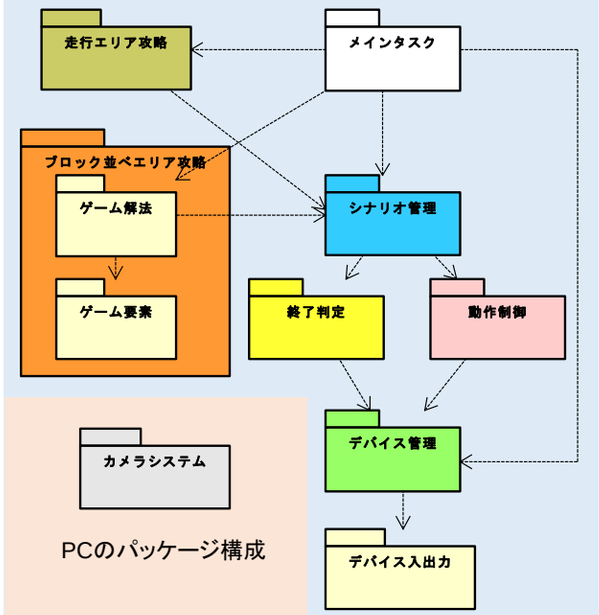
タスク設計
ブロックの位置・色を走行体が動作停止しているときにのみBluetooth通信を行う。これによりBluetooth受信と走行体の制御をタスクで優先度付けする必要はなく、1つのタスクで十分だと判断した。以上より、走行体側のタスクは分割しない設計とする。

3.2 設計概要(パッケージ図)

分析シートの解法や設計方針に従って、開発の分担や変更箇所が明確になるように役割に応じてパッケージ分割を行った。

パッケージ名	役割
カメラシステム	カメラからの画像情報をもとに、カラーブロック情報を抽出する。
走行エリア攻略	走行エリアを攻略するためのシナリオを生成する。
ブロック並べエリア攻略	ブロック並べエリアを攻略するためのシナリオを生成する。
ゲーム要素	ブロック並べゲームの要素について定義する。
解法	ブロック並べゲームを攻略するためのシナリオを生成する。
シナリオ管理	シナリオの管理・設定・切り替えを行う。
動作制御	シナリオで指定した動作定義に沿って、モータの制御量を決定する。
終了判定	シナリオで指定した終了条件を満たしているか判定する。
デバイス管理	デバイスからの値を加工した情報を管理する。
デバイス入出力	走行体に搭載しているセンサやモータと直接やり取りする。

走行体のパッケージ構成



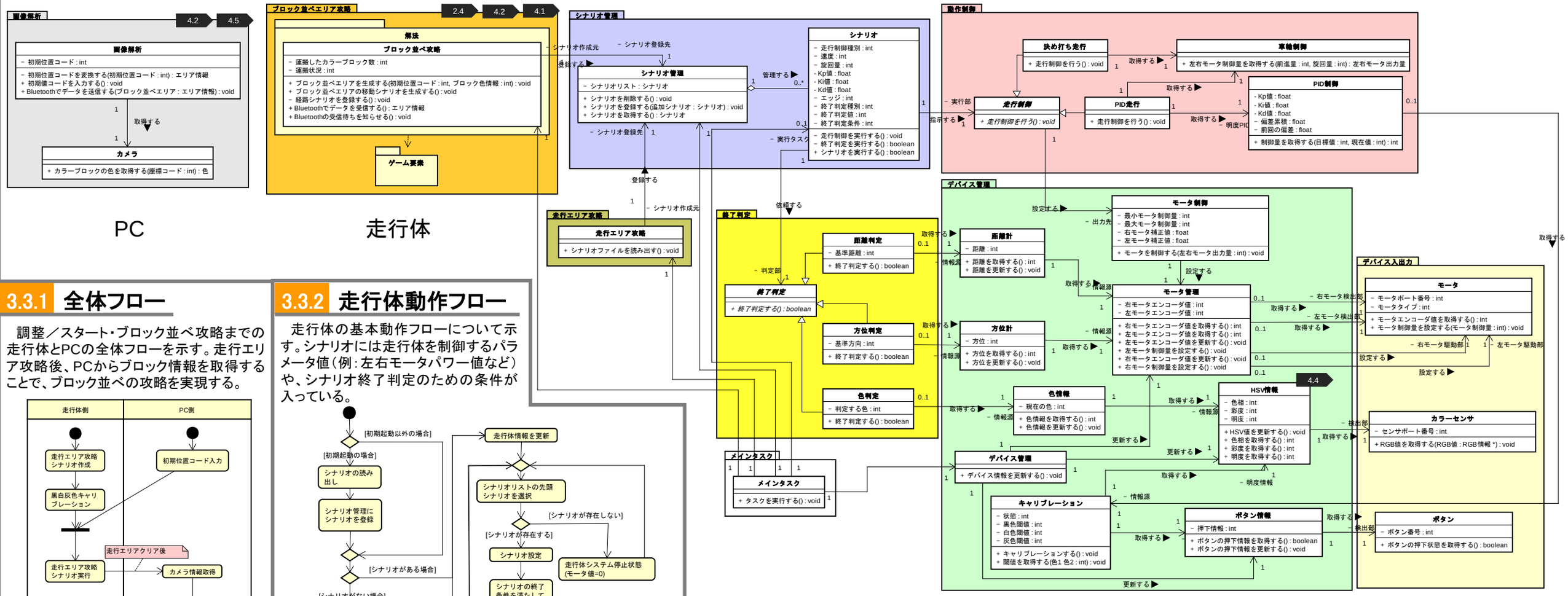
PCのパッケージ構成

3. 設計モデル(p.4)

FCT

3.3 設計詳細

ブロック並べ攻略に必要なクラスの詳細を示す。ブロック並べエリアで使用しないクラスなどについては一部省略している。
(例えばアームはプログラム起動時に初期位置を決定して以降は常にモータが動かないように固定してるだけなので省略している)



3.3.1 全体フロー

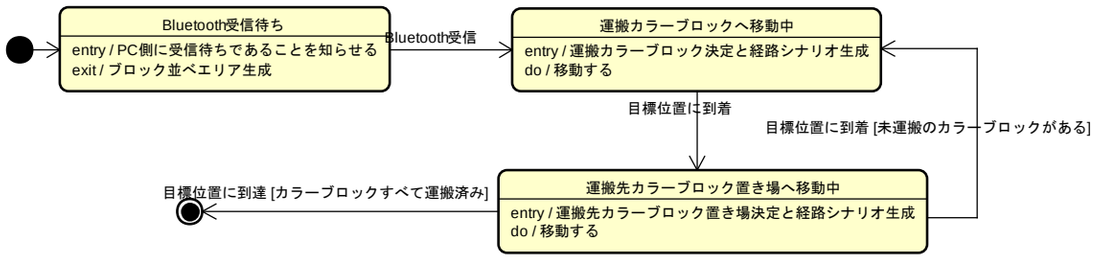
調整／スタート・ブロック並べ攻略までの走行体とPCの全体フローを示す。走行エリア攻略後、PCからブロック情報を取得することで、ブロック並べの攻略を実現する。

3.3.2 走行体動作フロー

走行体の基本動作フローについて示す。シナリオには走行体を制御するパラメータ値（例：左右モータパワー値など）や、シナリオ終了判定のための条件が入っている。

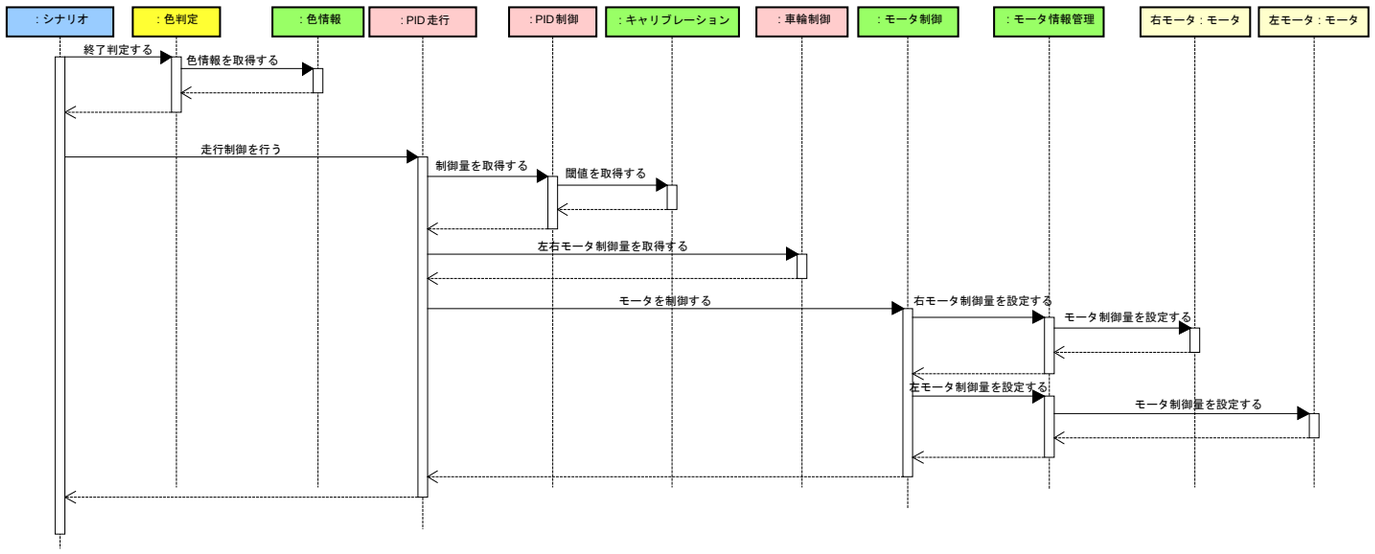
3.3.3 ブロック並べ攻略の状態遷移

ブロック並べ攻略のステートマシン図を示す。走行体は「運搬カラーブロックへ移動中」と「運搬先カラーブロック置き場へ移動中」で生成したシナリオをシナリオ管理に登録することで、ブロック並べエリア内を走行する。



3.3.4 走行体動作フロー

ブロック並べ攻略で生成されたシナリオとして特定の色までPID走行を実現するための処理シーケンスを示す。



4. 制御モデル(p.5)

ゲーム攻略とAIアンサー攻略に必要な機能を記述

FCT

制御戦略

4.1 ブロック並べエリア内走行

分析モデルで示した走行体のブロック並べエリアでの動作定義について、
それぞれどのような制御を行っているかを示す。
パターン①②③⑦⑧はカラーブロック置き場～線分の中間地点までの移動シナリオ
パターン④⑤⑥は線分の中間地点～線分の中間地点までの移動シナリオとしている。
理由：①②③⑦⑧の後に④⑤⑥を用いることでブロックをよけて移動を行うことができる。
パターン⑩は各パターンをつなぐ役割があり
各パターン終了地点～カラーブロック置き場の色を検知までの移動シナリオである。
パターン①～⑩を組み合わせることでブロック並べエリアを攻略することができる。

ポイント
・全ての動作は黒線の右エッジで始まり右エッジで終わるように統一することで連続動作の安定化と、パターン利用性を高めた。
・ブロックをブロック置き場に正確置く為に全パターン終了地点を線分の中間地点に統一し、ブロックを置く際は各パターン終了後にパターン⑩を用いてゆっくりライントレースを行い走行体を安定させる。
・旋回をする際は片輪止め走行でブロックすっぽ抜けを防止した。

図の表記説明

動作	説明	矢印
前進	前に進む	
後進	後ろに進む	
右旋	右旋回する	
左旋	左旋回する	
PID	PID走行で進む	

判定	説明
距離	距離による判定
方位	向きによる判定
色	色による判定

凡例

順番
動作
判定

パターン1：直進

①	②
前進	PID
距離	距離

パターン2：右折

①	②	③	④
右旋	右旋	右旋	PID
距離	黒色	白色	距離

パターン3：左折

①	②	③
左旋	左旋	PID
距離	黒色	距離

パターン4：直進

①	②	③	④	⑤	⑥
右旋	前進	左旋	前進	右旋	PID
方位	距離	方位	黒色	白色	距離

パターン5：右折

①	②	③	④	⑤
右旋	前進	右旋	右旋	PID
方位	黒色	方位	白色	距離

パターン6：左折

①	②	③	④	⑤
左旋	前進	前進	左旋	PID
方位	黒色	白色	黒色	距離

パターン7：180度旋回

①	②	③	④	⑤
前進	左旋	前進	右旋	PID
距離	距離	黒色	白色	距離

パターン8：180度旋回

①	②	③	④
後進	左旋	左旋	PID
距離	距離	白色	距離

パターン9：ブロック設置

①
後進
距離

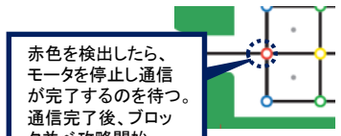
パターン10：つなぎ動作

①
PID
色

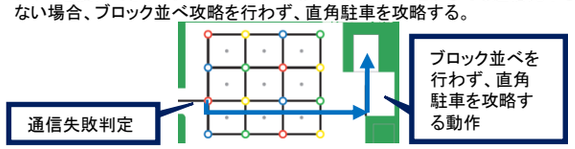
4.2 PCと走行体の通信

カメラシステムと通信することで走行体自身がアームを利用して色を取得するより短時間でブロックの色を取得可能。競技を有利に進められる。

PCと走行体の通信タイミング

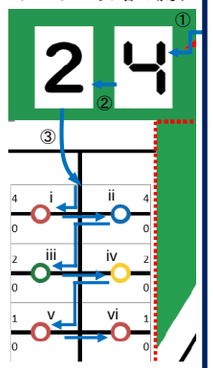


通信失敗時の動作
通信が完了しなかった場合、ブロック並べ攻略を開始できないのでボーナスタイムを獲得できず、タイムアップになる。一定時間が経過しても通信完了しない場合、ブロック並べ攻略を行わず、直角駐車を攻略する。



4.3 AIアンサー攻略

AIアンサー攻略の流れ



左／右出題数字判別
生成した左／右数字ビット列とあらかじめ測定しておいた0~7までの左／右数字ビット列（左／右数字マッチングデータ）の排他的論理和を算出する。算出結果のうち最も0が多いマッチングデータに対応する数字を解答数字とする。

（左／右数字ビット列）XOR（左／右数字マッチングデータ） ➡ 左／右解答数字

【右解答数字の導出例】

右数字マッチングデータ

0	00010
1	11011
2	01000
3	01001
4	10001
5	00101
6	00100
7	01011

右数字ビット列 00010 XOR

	0の数
0	5
1	2
2	3
3	2
4	2
5	2
6	3
7	3

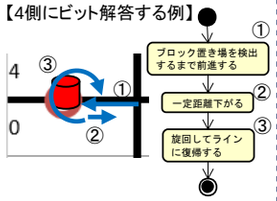
0の数が一番多い0が右解答数字として選ばれる

※左解答数字の導出もビット列の長さが異なるだけで同様の手順で導出する。

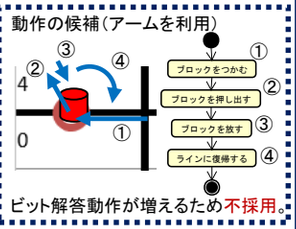
ライントレースをしてi~viの順に解答前ビット置き場に移動する。

ビット解答動作
動作の時間短縮、簡略化のため尻尾を利用する。二進数の0、1に応じて左／右旋回する。このときブロックに尻尾を当て、解答前ビット置き場からブロックを押し出す。

【4側にビット解答する例】



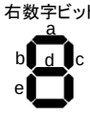
動作の候補（アームを利用）



ビット解答動作が増えるため不採用。

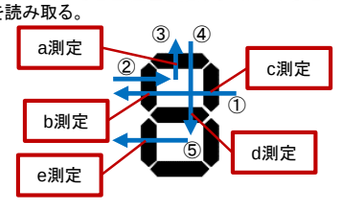
読み取り方の基本方針
・短時間で読み取る
・読み取り動作が少ない
・正確に数字を読み取る

右出題数字読み取り
a~eの各セグメントの状態（白：1、黒：0）をカラーセンサを用いて測定し、右数字ビット列を生成する。a~eのセグメントで数字を判別できるためすべてのセグメントは読み取らない。

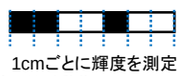


abcde
00000
右数字ビット列

読み取り動作
モータ角度から自己位置を推定し、a~eの各セグメントを読み取る。



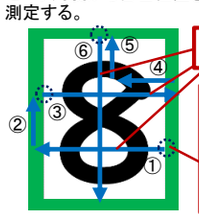
左出題数字読み取り
決め打ち走行で走行しながら距離計をもとに1cm間隔でカラーセンサの輝度（白：1、黒：0）を測定し、その値から左数字ビット列を生成する。



1cmごとに輝度を測定

0011011…
左数字ビット列

読み取り動作
モータ角度から自己位置を推定し①、③、⑥の輝度を測定する。



左数字の読み取り方候補
左出題数字の真ん中のみを読み取る方法が最も短時間かつ少ない動作で読み取りできそう。
➡ 3、5、6を混同することが判明したため不採用。

要素技術

4.4 色判定

課題　ブロック並べエリアを攻略しているとき自己位置判定だけでは誤差が大きくなってしまう。
対処方法　動作を切り替えるトリガに色判定を利用し、ブロック置き場を検出する。 RGB値をHSV色空間に変換することで、色の認識制度を上げる。

H(色相)

$$H=\begin{cases}60\times\dfrac{G-R}{S}+60, B=\min(R,G,B)\ のとき\\ 60\times\dfrac{B-G}{S}+180, R=\min(R,G,B)\ のとき\\ 60\times\dfrac{R-B}{S}+300, G=\min(R,G,B)\ のとき\end{cases}$$

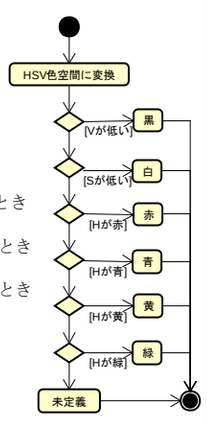
S(彩度)

$$S=\max(R,G,B)-\min(R,G,B)$$

V(明度)

$$V=\max(R,G,B)$$

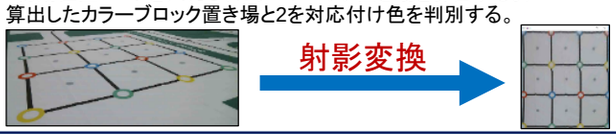
色判定手順



4.5 画像を用いたカラーブロックの色判別

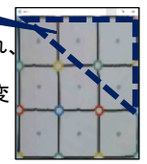
課題　限られた時間内に素早くブロックの色を判別し、運搬しなければならない。（アームを動かし、カラーセンサでブロックの色を判別する場合、1回の判別に2秒）
対処方法　カラーセンサではなくカメラを用いてブロックの色を判別し、ゲーム中に走行体が色を判別する動作を行わないことで、ゲームの攻略時間を短縮することが出来る。（カメラを用いた場合、色判別に必要な時間はほぼ0秒）

カラーブロックの色判別指針1
1. 射影変換を行い毎回同じ画像を取得する。射影変換とは、画像上の4点を指定し、形を変形する手法。今回はブロック並べエリアの四隅を指定し、長方形に変換した。
2. 16個のカラーブロック置き場に対応する画像上のx、y座標をあらかじめ指定しておき、カラーブロック置き場と対応付ける。
（3. 初期位置コードを入力し、カラーブロック置き場を算出する。）
4. 算出したカラーブロック置き場と2を対応付け色を判別する。

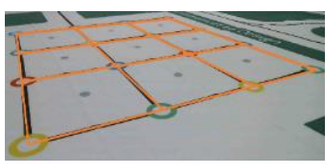


検証結果1　変換前画像の奥が手前と比べて小さいので引き延ばされ、画像が荒くなり色判別の精度が落ちた。
検証結果2　ブロック並べエリアの四隅を指定する際の少しのズレが変換後、大きなズレとなり、想定した位置の色が判別できない。
➡ カラーブロックの色判別指針1は不採用。

引き延ばされて画像が荒くなる。



カラーブロックの色判別指針2
1. カメラ取得画像上に格子状の四角形を描画し、線が交差する箇所のカメラ画像上のx、y座標を16か所指定しておく。
2. キャリブレーションタイム中にブロック並べエリアの格子とカメラ取得画像に描画された四角形が重なるようにWebカメラの向きを調整する。
（3. 初期位置コードを入力し、カラーブロック置き場を算出する。）
4. 算出したカラーブロック置き場とカメラ取得画像上の座標を対応付け色を判別する。



ブロック置き場に対応したカメラ取得画像上の座標(x,y)

検証結果1　2の操作の際、三脚の位置やカメラの台座の調整が難しい。
検証結果2　コースが歪んでいるとブロック並べエリアと四角形が重ならない場合があり、正確な位置の色の取得が困難。
➡ カラーブロックの色判別指針2は不採用。
最終的な指針として、画像処理を行わずカメラ取得画像上のブロック置き場16か所を直接指定する方法を考えた。

カラーブロックの色判別指針3
1. カメラ取得画像上に写っている16か所のカラーブロック置き場の場所をクリックし、各ブロック置き場に対応したx、y座標を指定。
（2. 初期位置コードを入力し、カラーブロック置き場を算出する。）
2. カメラシステムは走行体から要求が来るまで待機。
（3. 走行体がブロック並べエリアに到達するとブロックの初期位置とブロックの色情報を送信するようカメラシステムに要求する。）
4. カメラシステムは手順2で算出したカラーブロック置き場に対応する座標の色を判別する。
（5.カメラシステムは走行体にブロックの初期位置と、ブロックの色を送信する。）