

1. 要求

2. 分析

3. 制御

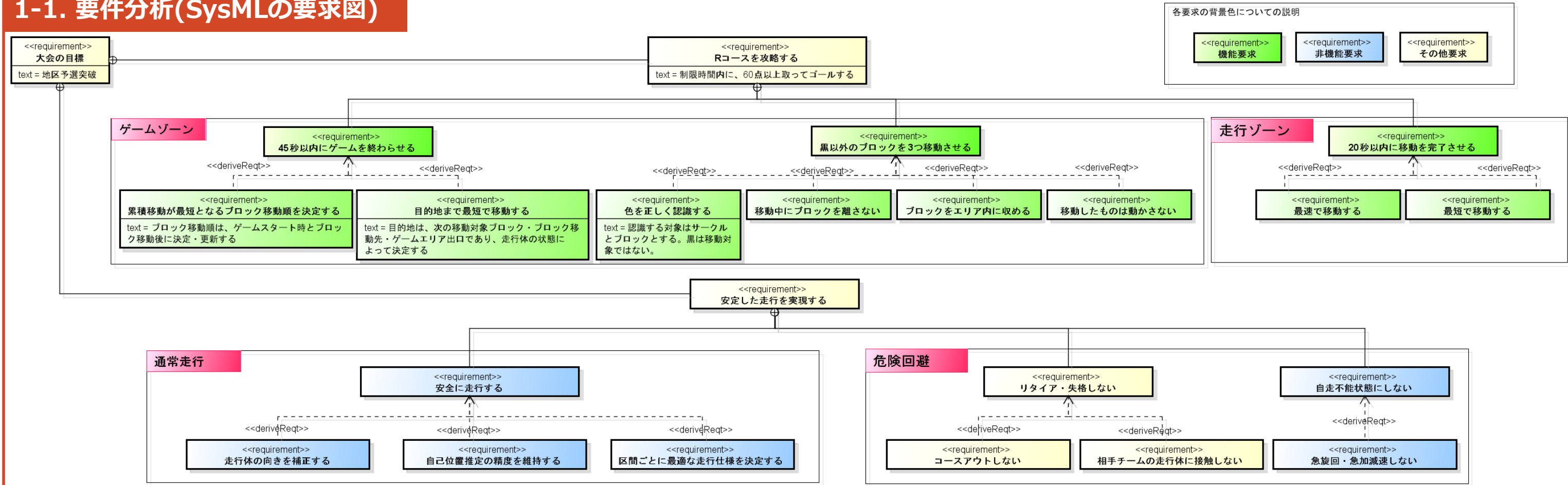
4. 設計(1)

4. 設計(2)

しんよこ
(°Д°)

Rコースで60点以上取ることを目標とし、要求事項を整理した

1-1. 要件分析(SysMLの要求図)



1-2. 要求に対応する要素

要求	要求に対応する要素と参照先										
	(移動対象ブロック決定 ゲームの攻略手順)	ブロックの色の管理	自己位置推定機能	経路の選択基準	座標指定移動	最適速度設定	色認識	アーム制御	向き補正走行	自己位置補正	ゲームエリアと走行体の抽象化
	2-6	2-5	3-1	2-4	3-3	3-6	3-5	3-5	3-4	3-2	2-1
累積移動が最短となるブロック移動順を決定する	●	●									
目的地まで最短で移動する			●	●	●	●					●
色を正しく認識する							●	●			
移動中にブロックを離さない					●	●					
ブロックをエリア内に収める					●	●					
移動したものは動かさない		●									
最速で移動する						●					
最短で移動する				●	●	●					●
走行体の向きを補正する									●		
自己位置推定の精度を維持する									●	●	●
区間ごとに最適な走行仕様を決定する											●
コースアウトしない											
相手チームの走行体に接触しない											
急旋回・急加減速しない											

※要求に対応する要素の詳細は後述する
(記述先は各番号を参照)

地区予選突破に、Rコースで必要となる点数を60点以上と想定し、チームの目標とした。

目標達成に必要な要求をSysMLの要求図を用いて段階的に詳細化した。(1-1. 要件分析)

また、要件分析で抽出した派生要求を実現するため、各要求に対して、必要となる要素技術を洗い出した。(1-2. 要求に対応する要素技術)

1. 要求

2. 分析

3. 制御

4. 設計(1)

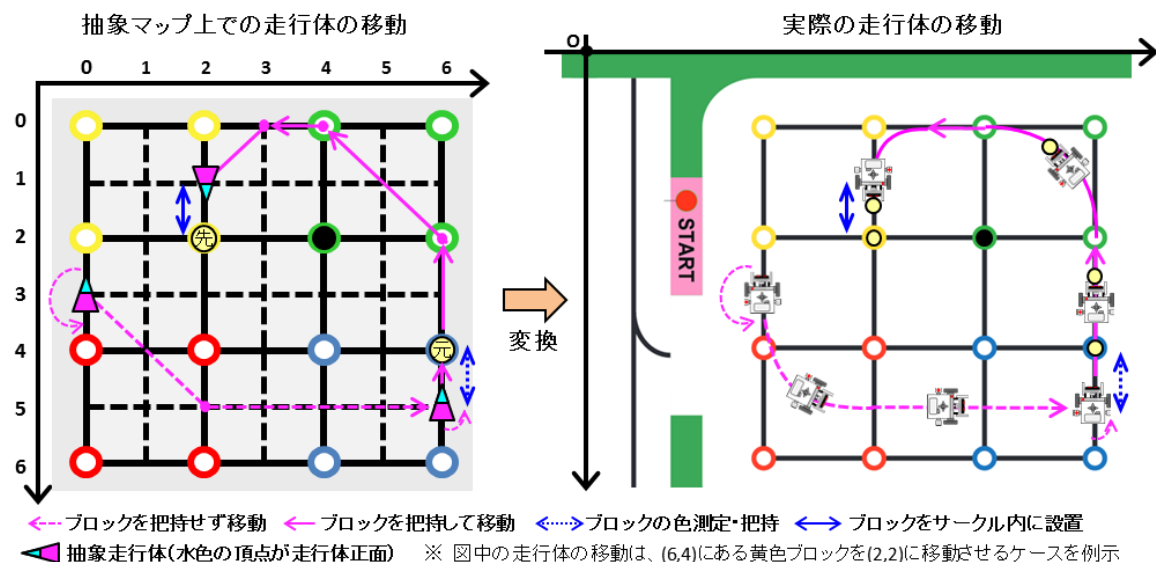
4. 設計(2)

しんよこ
(° D°)

抽象化したゲームエリア上での走行体の振る舞いと実際の操作を分離

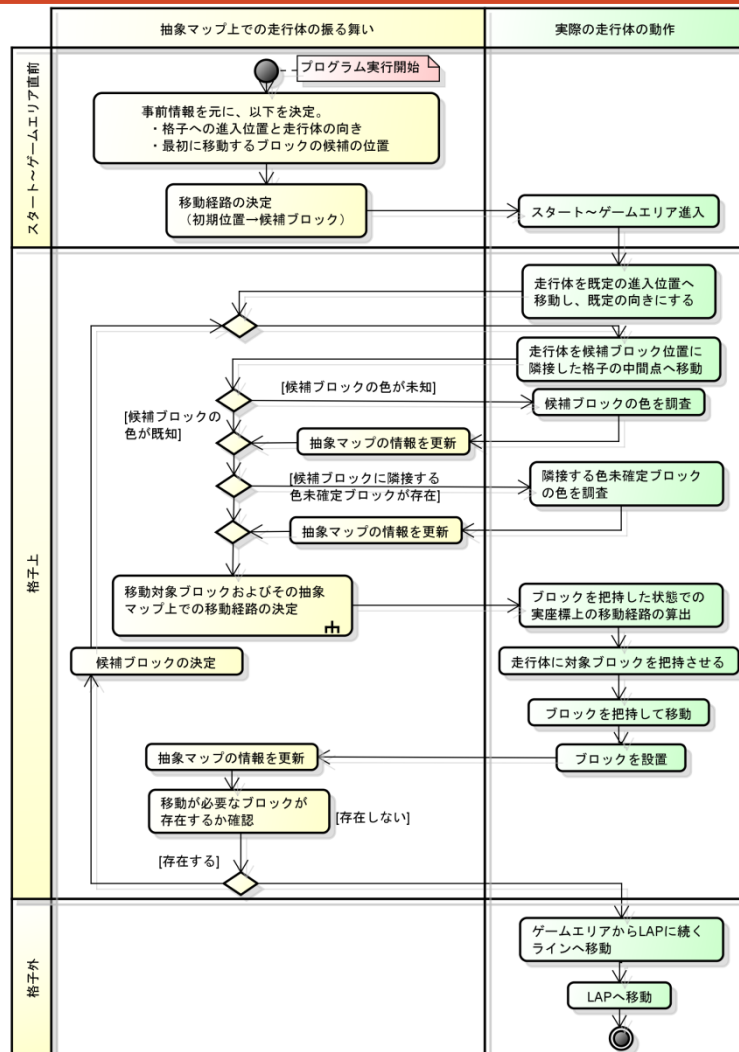
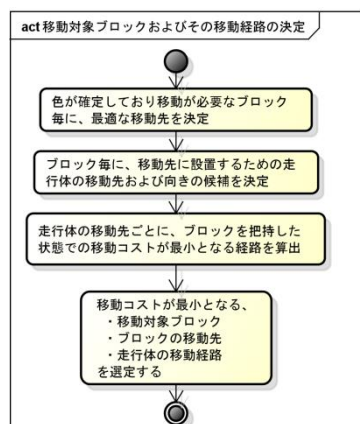
2-1. ゲームエリアと走行体の抽象化

4 x 4 マスの格子であるゲームエリアを、格子点間の midpoint で分割し、下図のような座標系で表現した**抽象マップ**を設ける。抽象マップ上では走行体は位置、向き、把持ブロックの有無を持つ**抽象走行体**として表現する。走行体の振る舞いを抽象マップ上で検討し、実際の動作へ変換する。



2-6. ゲームの攻略手順

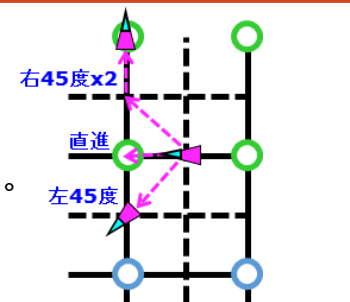
2-1～2-5で定義したモデルを用いてゲームを攻略するための手順を示す。抽象マップから実際の座標系への変換および走行体の各種動作については、**3. 制御**で述べる。



2-2. 抽象走行体の移動規則

抽象走行体の移動は以下の規則に従う。

1. 停止中は、常にサークルとサークルの間(以降、**中間点**)に存在する。
2. 停止中の向きは、必ずXまたはY軸に平行である。
3. ブロックを**把持しておらず、かつ停止中のみ**左右**90**または**180**度向きを変えることができる。
4. **移動中かつ格子点上でのみ**その向きを左右**45**度変えることができる(左図に例示)。
5. ブロックを把持する際を除き、ブロックが設置されている点を通過しない。



2-3. 抽象マップ上での経路探索

1. 探索範囲および経路点の設定 始点から終点への経路の探索範囲は、『始点』と『終点』での走行体の向きと反対の隣接点(右図の**点C**)を中心とする**最大2x2**の矩形(以降、**終点近傍**：右図の**赤破線**)を含む矩形(右図の**青破線**)とする。走行体は探索範囲中の格子点を経由するが(以降、**経路点**：右図の**ピンク色の点**)、始点と終点を除く矩形の頂点は除外する。

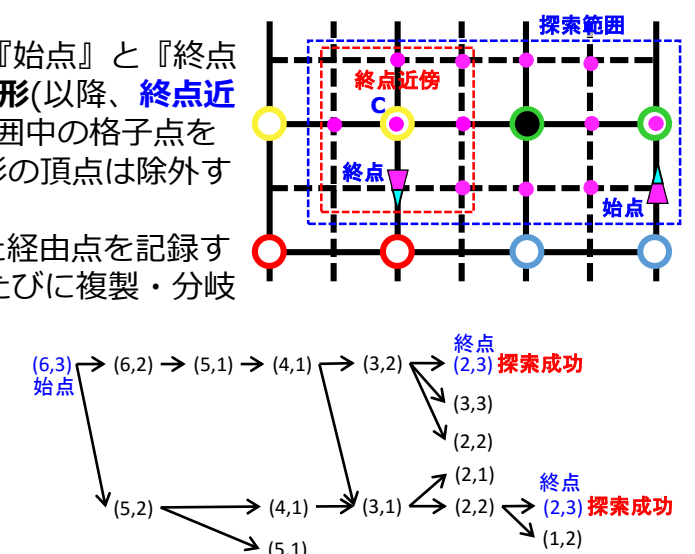
2. 経路の探索 始点を出発し**2-2**の規則に従って移動しながら、通過した経路点を記録する**探索子**を設ける。探索子は右下図のように、次の移動先が複数になるたびに複製・分岐し、それぞれの方向に移動する。探索子の停止条件は以下の通り。

(探索成功)

- ・終点に到達し、終点での走行体の向きとの差が±45度以内である。

(探索失敗)

- ・終点に到達し、終点での走行体の向きとの違いが±90度以上である。
- ・点Cを除く**終点近傍**の点を2回通過する。
- ・同じ経路点を2回通過する。
- ・移動可能な経路点が存在しない。



ブロックを把持した場合の探索子の分岐パターン例

2-4. 経路の選択基準

2-3で探索した経路の、抽象マップ上での移動距離と抽象走行体の向きの変更から**移動コスト**を算出し、移動コストが最も低い経路を選択する。

1. 移動距離由来のコスト 経路上の隣接する経路点を結ぶ直線の長さの総和。

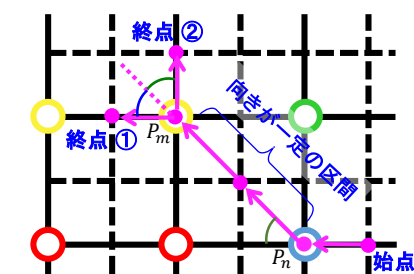
2. 向きの変更に由来するコスト

抽象走行体の移動	現実の走行体への影響
向きの急激な変化	移動速度低下, 自己位置推定の精度低下
向きの変化に要した移動距離	向きの急激な変化による現実の走行体への影響を低減

経路上で向きの変化する点: P_n 点 P_m の次に向きが変化する点: P_m
点 P_n での向きの変化: A_n (度) 点 P_m での向きの変化: A_m (度) ... (向きの変化は右回りを正とする)

$$\text{向き変更に由来するコスト} = \sum \frac{2(1 + |A_n - A_m|/90)}{|P_n P_m|}$$

移動コスト = 移動距離由来のコスト + 向きの変更に由来するコスト

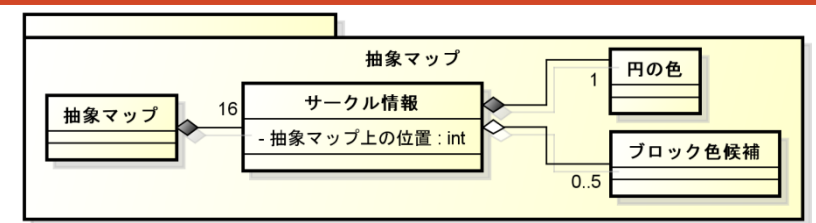


(例) 上図での①の移動コスト
移動距離由来のコスト = $1 + \sqrt{2} \times 2 + 1 \approx 4.82$
向きの変更に由来するコスト = $2 \times (1 + \frac{|45+45|}{90}) / (2\sqrt{2}) = 1.41$
移動コスト = $4.82 + 1.41 = 6.23$

2-5. ブロックの色の管理

抽象マップ上では、各サークル

は自身の色情報と設置されているブロックの色の候補を0~5個持つ(色候補が0個のサークルはブロックが設置されていない)。ブロックの色が確定、またはブロックが移動される度に抽象マップの情報は更新される。



1. 要求

2. 分析

3. 制御

4. 設計(1)

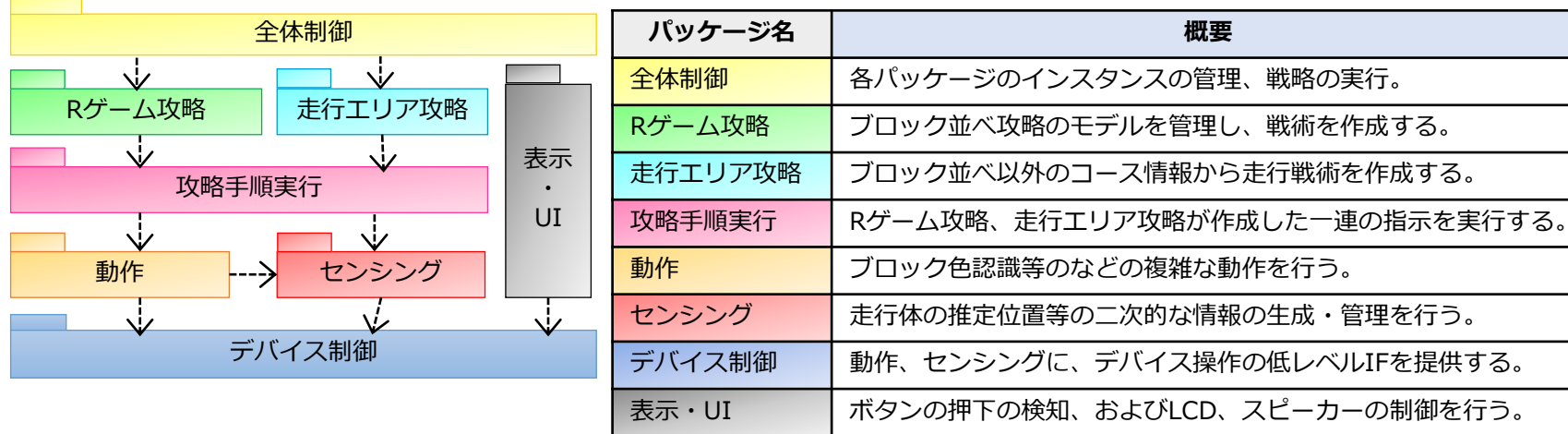
4. 設計(2)

しんよこ
(°Д°)

ブロック並べのドメインモデルと走行体の操作を分離する

4-1-1. パッケージ構成

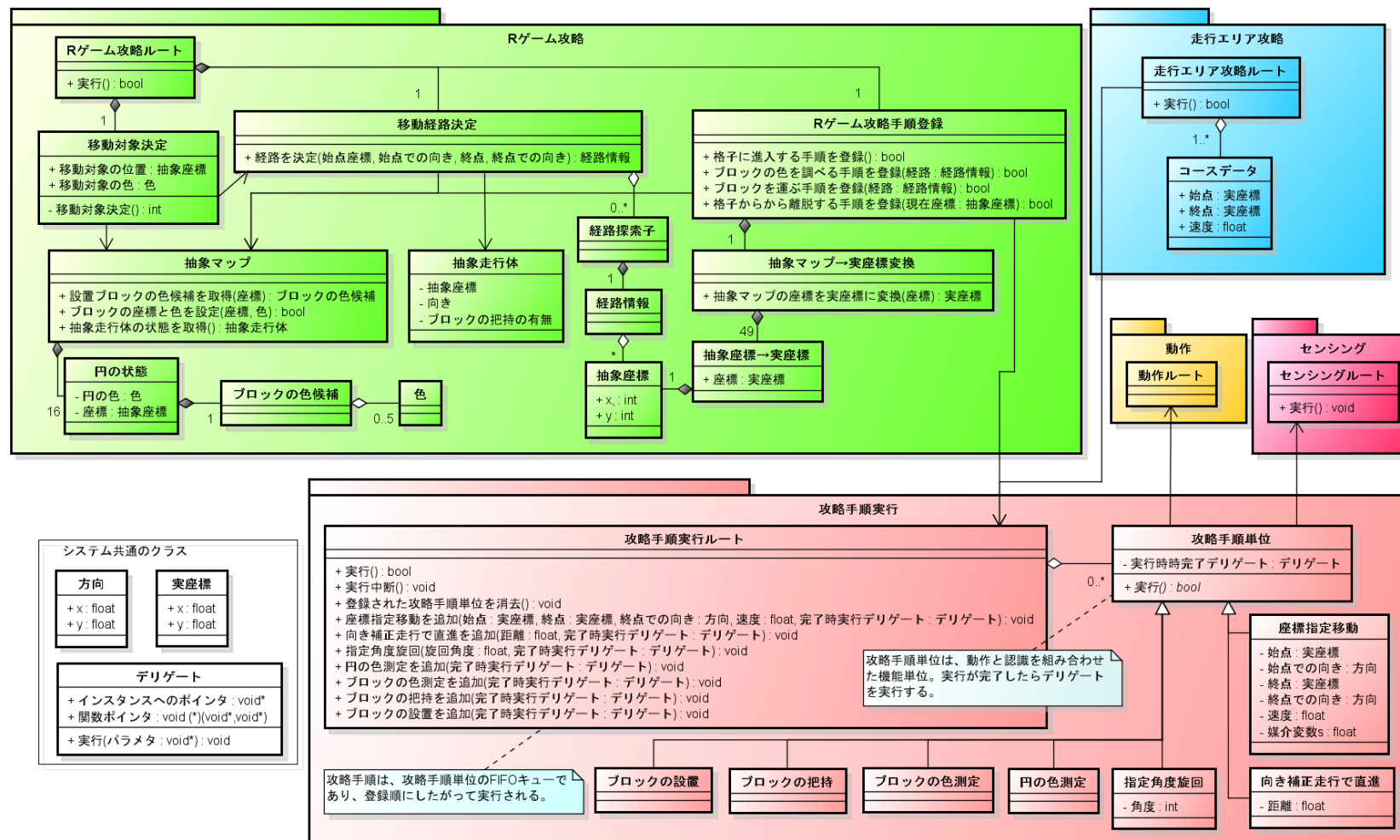
システムをその機能の抽象度に着目して階層化し、パッケージとした(表示・UIを除く)。上位層の機能はその直下の層によって実現される。



4-1-3. Rゲーム攻略・走行エリア攻略・攻略手順実行

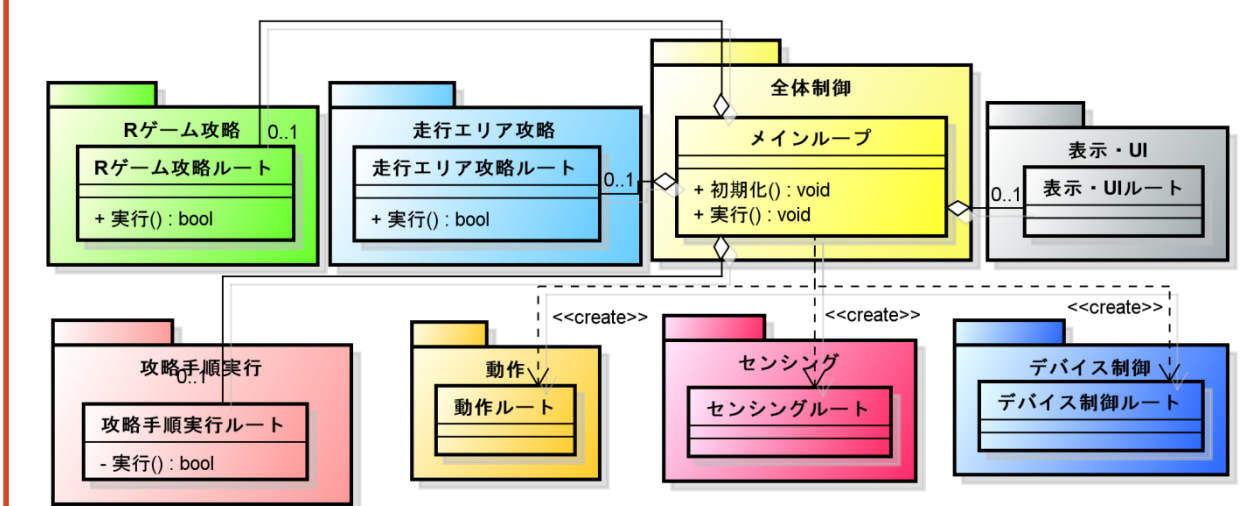
Rゲーム攻略および走行エリア攻略は直接走行体を

操作せず、攻略の手順を攻略手順実行に登録する。攻略手順実行は登録された順に攻略手順を実行する。攻略手順実行の結果(ブロックの色測定等)は、Rゲーム攻略の状態更新メンバ関数のデリゲートを通じて反映される。



4-1-2. 全体制御

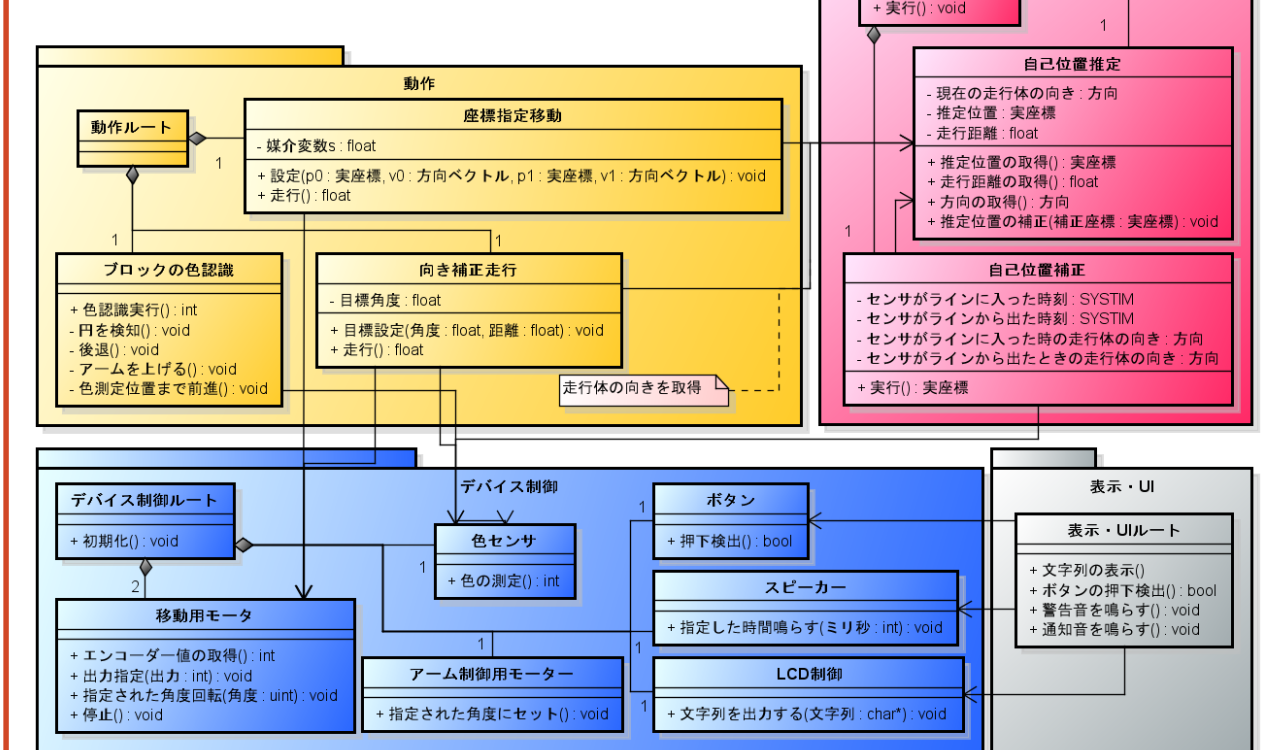
全体制御は各パッケージのルート(外部からパッケージを利用する際の窓口となるオブジェクト)を作成する。また、Rゲーム攻略、走行エリア攻略、表示・UIのルートのインスタンスを保持する。メインループ内では、Rゲーム攻略または走行エリア攻略および攻略手順実行パッケージのルートの実行関数が呼び出される。



4-1-4. 動作・センシング・デバイス制御・UI

動作およびセンシングは、攻略手順実行が必要とする機能(3.制御)を提供する(例: ブロック色認識、自己位置推定等)。これらの機能はEV3 APIの簡易的なラッパーであるデバイス制御を用いて実現される。

表示・UIは走行の開始時およびデバッグでのみ使用される機能で、他のパッケージからユーティリティ的に使用される。



1. 要求

2. 分析

3. 制御

4. 設計(1)

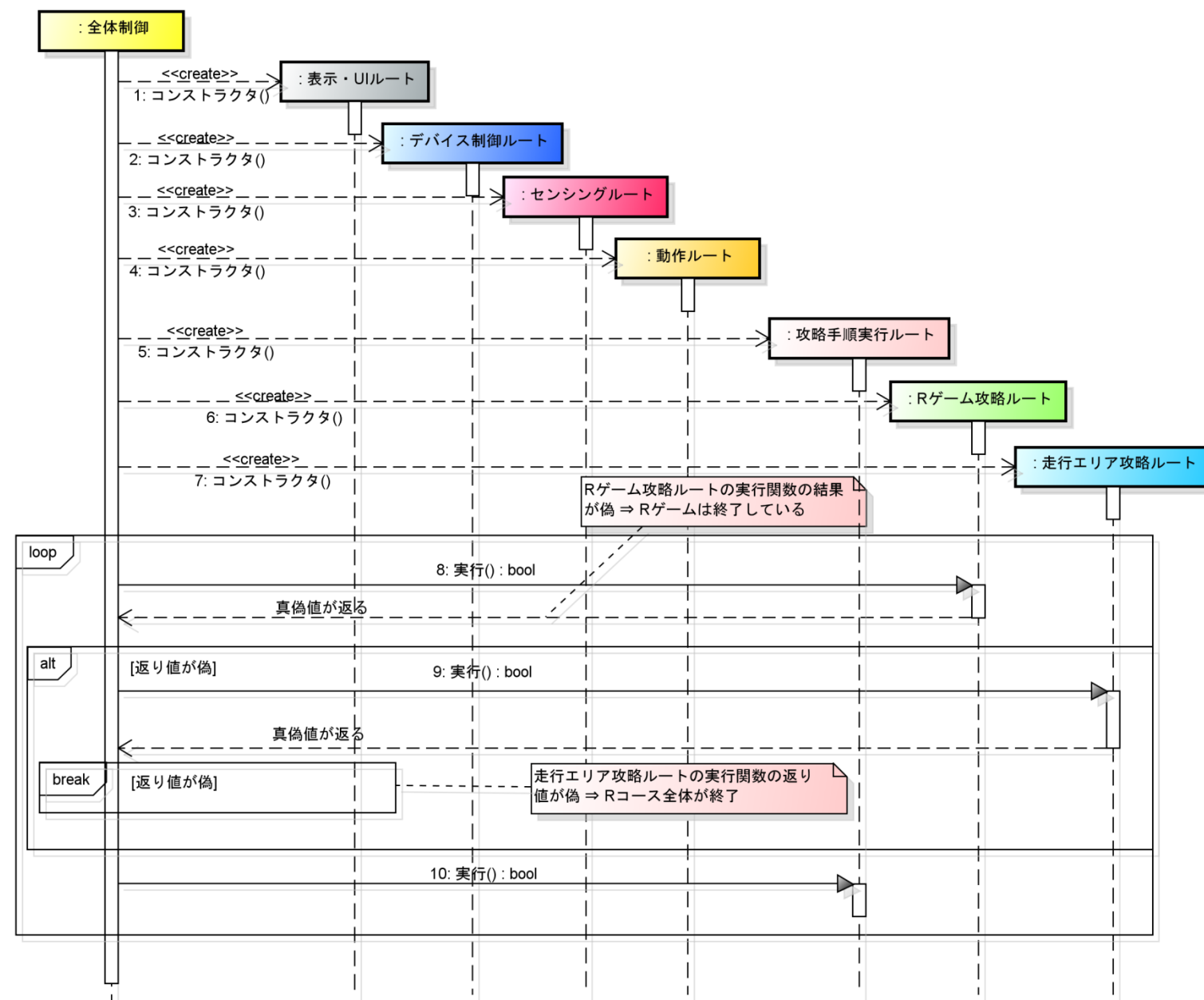
4. 設計(2)

しんよこ
(°Д°)

システムは各パッケージのメイン関数の呼び出しで実行される

4-2-1. システム全体のシーケンス

プログラム開始時に、**4.1** で示した階層の下から上の順に全パッケージのルートが作成される。全パッケージのルートを作成後、全体制御のメインループで、**Rゲーム攻略ルート**、**走行エリア攻略ルート**、**攻略手順実行ルート**の実行関数が呼びだされる。



4-2-2. Rゲーム攻略のシーケンス

Rゲームの攻略をシーケンスを、攻略手順の作成と実行に注目して例示する（各パッケージのルートは既に作成済み）。

