

付録 B

デザインパターン ライブラリ

B.1

スタティックエンジン

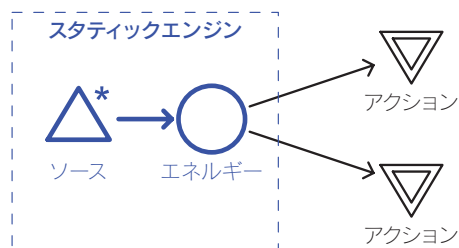
- **種類**：エンジン
- **意図**：プレイヤーがゲーム中に消費または収集するリソースを、安定したフローにより生産する。
- **動機**：スタティックエンジンは常に供給される安定したリソースフローを作り出す。

適用可能性

スタティックエンジンの使用場面：

- デザインを複雑化せずにプレイヤーのアクションを制限したい。スタティックエンジンをを使うと、プレイヤーは、長期計画の必要性はないものの、リソースの使い方を考える必要にせまれる。

構造



パーティシパント

- スタティックエンジンが生産する**エネルギー**
- エネルギーを生産する**ソース**
- プレイヤーがエネルギーを費やす対象の**アクション**

コラボレーション

ソースは固定の速度 (rate) または予測不能な速度でエネルギーを生産する。

結果

スタティックエンジンの生産速度は不変であるため、ゲームのバランスへの影響は極めて予想可能である。スタティックエンジンによってバランスが崩れるのは、プレイヤー全員を同一の生産速度にしない場合のみである。



メモ

スタティックエンジンは、リソースを費やして得られる複数の選択肢をプレイヤーに提供する必要がある。1つしか選択肢のないスタティックエンジンはあまり使い道がない。スタティックエンジンからリソースが収集できることは、誰の目から見ても明らかであるから、一般に長期的な戦略をもたらすきっかけにはならない。

実装

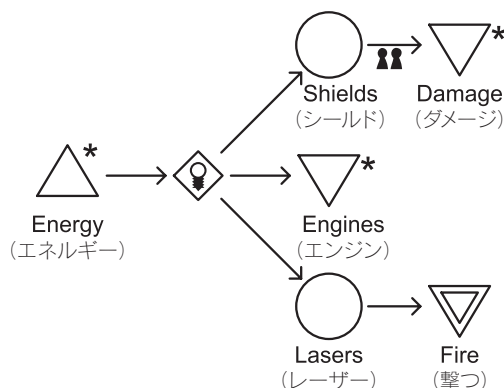
通常、スタティックエンジンの実装はシンプルである。エネルギーを生産するシングルソースで十分だ。エネルギーの生産に複数の段階を含めることは可能だが、そうしたとしても一般的にゲームにはほとんど変化がない。

生産速度に何らかの形式で変化を持たせれば、スタティックエンジンを予測不能にできる。予測不能なスタティックエンジンによって、プレイヤーはリソースが少なくなる期間のために備えをする必要にせまられたり、逆境にもちこたえる計画的なプレイヤーにリワードを与えたりできる。最も簡単に予測不能なスタティックエンジンを構築するには、ランダム性を使ってリソースの出力量や生産しない期間の長さを変化させることだが、スキルやマルチプレイヤーのダイナミクスによっても予測不能にできる。

ランダムな生産速度の結果を各プレイヤーで同一にすることは可能だが、必ずしもその必要はない。全プレイヤーに同じリソースを生産する予測不能なスタティックエンジンを使うと、予測不能性に影響することなく運に関わる係数が均等になる。これは、このパターンがもたらす計画性やタイミングをより重要にする。例として、各プレイヤーが自分の獲得できるリソース数をひそかに決めるゲームが挙げられる。最小数は、全員がプレイを開始するのに必要なリソース数になり、最小数を決めたプレイヤーが最初にプレイできる。これによって、ゲームの現在の状態からこのメカニクスに何らかのフィードバックを自動的に設定できる（このシステムはインフレーションを阻む働きをする）。

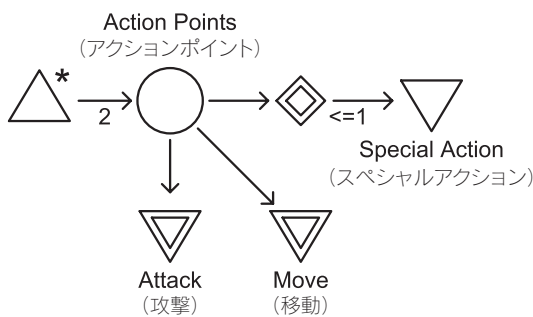
例

スタティックエンジンの例として、Star Wars : X-Wing Allianceの宇宙船が生産するエネルギーが挙げられる。このエネルギーを変換すると、プレイヤーのシールドやスピード、レーザーを増強できる。このゲームでは戦略上極めて重要な決断であり、エネルギーの配分はいつでも変更できる。毎秒生産されるエネルギー量は、同型の宇宙船ではすべて同じである（図B.1）。



図B.1 Star Wars: X-Wing Allianceにおけるエネルギー配分

多くのターン制ゲームにおける、ターンごとの限られた回数のプレイヤーアクションも、スタティックエンジンと見なすことができる。つまり、アクションの選択がゲームの中心になり、アクションを後のターンのために温存しておくことはできない。ファンタジーのボードゲーム Descent: Journeys in the Dark ではこのメカニズムが使われている。ターンごとに、プレイヤーは自分のヒーローの行える3つのアクション（移動、攻撃、特別アクションの準備）の中から1つを選択できる（図B.2）。この図では、ターンごとにプレイヤーは2つのアクションを選択できるが、特別アクションはターンごとに1回しか行えない。これによって、攻撃を2回、移動を2回、攻撃と移動、攻撃と特別アクションを行う、移動と特別アクションを行う、という5つの組み合わせが可能となる。



図B.2 ボードゲームDescent: Journeys in the Darkのアクションポイントの配分

関連するパターン

- 弱いスタティックエンジンはコンバータエンジンのデッドロック状態を防ぐことができる。
- スタティックエンジンは、ダイナミックエンジン、コンバータエンジン、またはスローサイクルパターンによって精緻化できる。

B.2

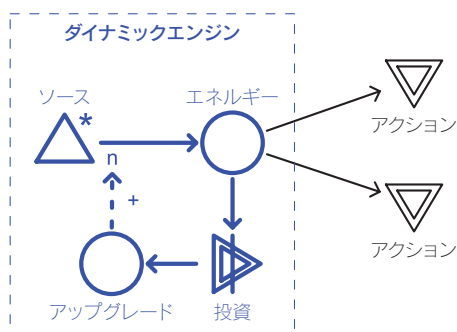
ダイナミックエンジン

- **種類**: エンジン
- **意図**: 調整可能なフローを持つリソースを生産する。プレイヤーはリソースを投資に回してフローを調整できる。
- **動機**: ダイナミックエンジンは安定したフローのリソースを生産する。また、プレイヤーはリソースを使って生産を向上させることができるため、長期的な投資の可能性をもたらす。ダイナミックエンジンのコアはポジティブな建設的フィードバックループである。

適用可能性

ダイナミックエンジンを使うと、長期的な投資と短期的なゲインの間のトレードオフを導入できる。ダイナミックエンジンはスタティックエンジンよりも多くのコントロールをプレイヤーに与える。

構造



パーティシパント

- 他のエンジンによって生産される**エネルギー**
- エネルギーを生産する**ソース**
- エネルギーの生産速度に影響を与える**アップグレード**
- アップグレードを生み出す**投資アクション**
- プレイヤーがエネルギーを費やす対象の**アクション**。「投資」アクションを含む

コラボレーション

ダイナミックエンジンは、多くのアクションによって消費されるエネルギーを生産する。この中の1つのアクション（投資）は、ダイナミックエンジンのエネルギーの出力を向上させるアップグレードを生産する。ダイナミックエンジンでは、プレイヤーが生産を向上させるために投資できる2種類のアップグレードがある。

- エネルギーを生産する**頻度**
- 一度に生産される**エネルギートークン数**

この2つには微妙な違いがある。頻度を高めると安定したフローを生み出し、逆に（頻度は低いまま）トークン数を多くするとエネルギーの爆発的増加を生み出す。

結果

ダイナミックエンジンは、強力かつポジティブな建設的フィードバックループを構築する。この強力なループは、何らかの摩擦などのネガティブフィードバックを実現するパターンを使って、バランスをとる必要があるだろう。代わりの方法としては、難易度を増大させるチャレンジのエスカレーションを使ってバランスをとることもできる。

ダイナミックエンジンを使う際、長期的戦略を優先させすぎたり、逆に長期的戦略のコストを上げすぎたりして、絶対優位の戦略を生まないように気をつけることが重要である。

ダイナミックエンジンによって独特のゲームプレイ体験が生まれる。ダイナミックエンジンしかないようなゲームでは、当初、プレイヤーは投資ばかりを行い、あまり進展がないように見える。ある地点を超えるとプレイヤーに進展が見え始めるので、プレイヤーは可能な限りその状態へ持っていく必要がある。

実装

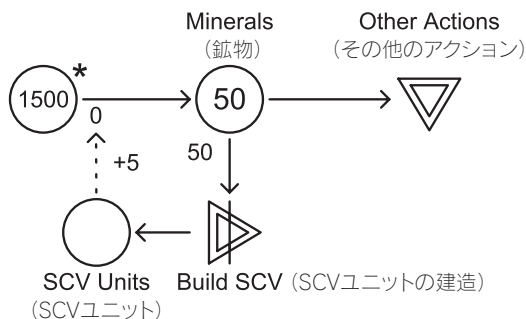
長期的投資または短期的投資が絶対優位の戦略として浮上する確率は、ダイナミックエンジンに何らかのランダム性を導入すれば削減できる。とはいえ、予測不能なダイナミックエンジンに存在するポジティブフィードバックループによって、プレイヤーのゲーム序盤の運が増幅され、その結果、ただちにランダム性が高くなりすぎる可能性がある。

ランダムな生産速度の結果をプレイヤー全員に対して同じにすることは可能だが、必ずしもその必要はない。全プレイヤーに同じリソースを生産する予測不能なダイナミックエンジンを使うと、予測不能性に影響することなく運に関わる係数が削減する。

これによって、プレイヤーが選択した戦略がより重要になる。アップグレードをエネルギーに戻す変換を許すダイナミックエンジンもあり、一般に元の投資よりもレートが低い場合に使われる。アップグレードが高価で、プレイヤーが頻繁に大量のエネルギーを必要とする場合、これは有効な選択肢である。

例

StarCraftで宇宙建設車両 (SCV) ユニットの能力の1つである鉱物の採掘能力は、新たにSCVユニットを構築するのに費やして高めることもできる (図B.3)。要するに、SCVユニットはゲームを推進するダイナミックエンジンである (もっともStarCraftでは鉱物の量は限られており、SCVユニットも敵に破壊される可能性がある)。直接的には、長期的な選択肢 (多くのSCVユニットに投資する) と短期的な選択肢 (軍事ユニットに投資して素早く敵を攻撃または直近の脅威に対応する) をプレイヤーに与えている。



図B.3 StarCraftでの鉱物の採掘

「カタンの開拓者たち」の経済は、確率に左右されるダイナミックエンジンを中心に展開する。ダイスを振ることで、各プレイヤーのターン開始時にリソースを生産するゲーム盤のタイル(土地)が決まる。村をたくさん作れば作るほどターンごとに受け取るリソース数が増える。また、村を町にアップグレードすると、タイルごとのリソース出力が2倍になる。「カタンの開拓者たち」は、さまざまな種類の投資アクションを許し、エネルギーではなくアップグレードを測って勝者を決定することで、ダイナミックエンジンが作成してしまいがちな典型的な傾向を回避している。「カタンの開拓者たち」についての詳細な解説と図は、第11章、281ページの「進行を間接的に生成する」の項を参照してほしい。

関連するパターン

- ダイナミックエンジンとアトリションは、ダイナミックエンジンの長期的な利益を相殺するのに適切なパターンである。その一方で、静摩擦は長期的な投資を際立たせる。
- ダイナミックエンジンはスタティックエンジンパターンを精緻化する。
- ダイナミックエンジンは、エンジン構築やワーカー配置パターンによって精緻化できる。

B.3

コンバータエンジン

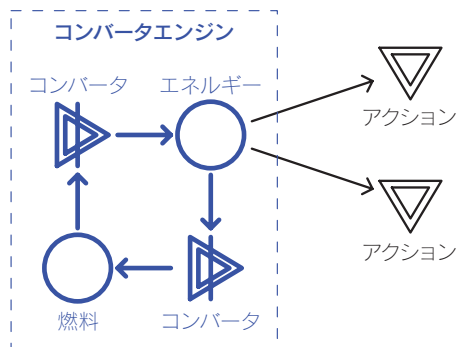
- **種類:** エンジン
- **意図:** 2つのコンバータでループを形成して、ゲームの他の部分で使用できる余剰のリソースを生産する。
- **動機:** 互いに変換可能な2つのリソースによって、余剰のリソースを生産するフィードバックループに拍車をかける。最低でも1つのコンバータが、入力より出力を多くして余剰のリソースを生産する必要がある。コンバータエンジンは、他の大半のエンジンよりも複雑なメカニズムであり、エンジンを向上させる多くのチャンスを提供する。結果的に、コンバータエンジンはほぼ常に動的である。

適用可能性

コンバータエンジンの使用場面：

- スタティックエンジンやダイナミックエンジンが提供するよりも多くのリソースをプレイヤーに提供できるよう、より複雑なメカニズムを構築したい（ここでのコンバータエンジンの例では、インタラクティブな要素が2つあるが、ダイナミックエンジンには1つしかない）。フィードバックループの強さと必要な投資を見定めるのがずっと難しくなるため、ゲームの難易度が上がる。
- エンジンを駆動するフィードバックループのプロファイルと、それによってゲームへ流れるリソースのストリームをチューニングするために、複数の選択肢やメカニクスが必要である。

構造



パーティシパント

- 2つのリソース：エネルギーと燃料
- 燃料をエネルギーに変換するコンバータ
- エネルギーを燃料に変換するコンバータ
- エネルギーを消費するアクション

コラボレーション

2つのコンバータは燃料をエネルギーに、またエネルギーを燃料に変換する。一般にプレイヤーは、最終的にスタート時よりも多くのエネルギーを持つことになる。

B

結果

コンバータエンジンはデッドロック状態を引き起こす可能性がある。リソースが2つとも枯渇するとエンジンは停止する。燃料を得るためにエネルギーの投資を忘れると、プレイヤーは自らデッドロックを引き起こすリスクがある。これを回避するには、コンバータエンジンと、弱いスタティックエンジンを組み合わせる。

コンバータエンジンはプレイヤーにスキルを要求する。コンバータを手動で操作する必要がある場合は特にそうだ。

ダイナミックエンジンと同様に、ポジティブフィードバックループがコンバータエンジンを駆動する。たいていの場合、このフィードバックループは何らかの摩擦を適用してバランスをとる必要がある。

実装

コンバータエンジンのフィードバックループのステップ数は、効率的に操作する上での難易度に大きく影響する。ステップ数が多くなると難易度が増大し、逆に少なくなると減少する。同時に、ステップ数が多いと、エンジンへの調整や追加を行う機会が増える。

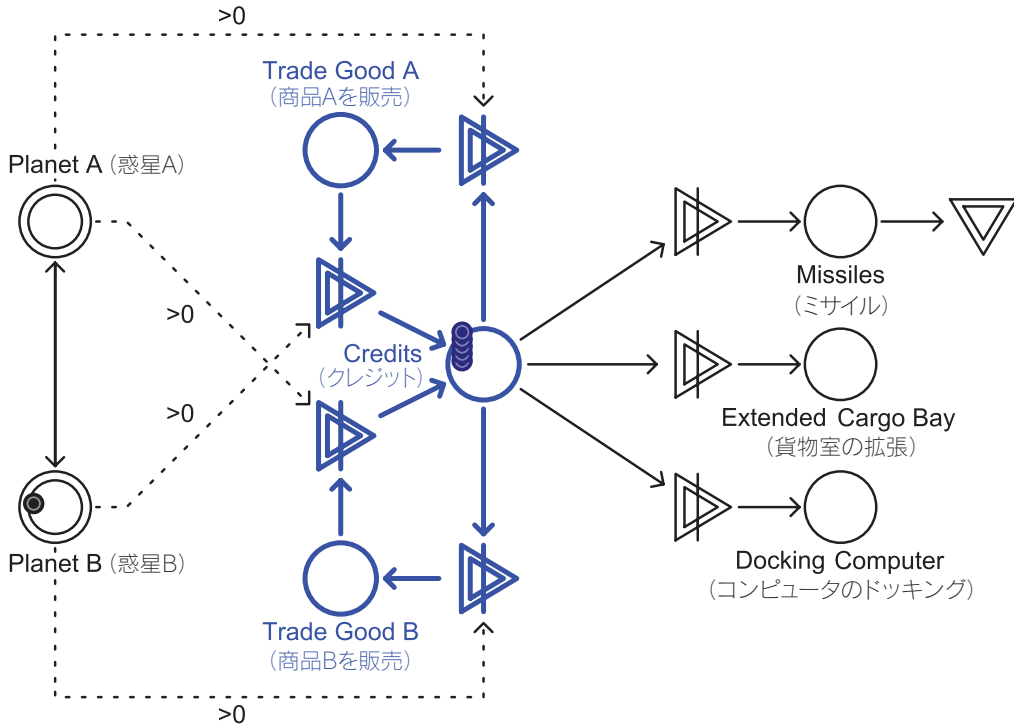
ステップ数が少なすぎるシステムでは、コンバータエンジンのメリットが限られるので、むしろダイナミックエンジンを使うことを検討したほうがよい。ステップ数が多すぎると、操作や保守が面倒になる。エンジンのさまざまな要素を自動化できないボードゲームでは特にこの傾向が強い。

ランダム性やマルチプレイヤーのダイナミクス、スキルなどをフィードバックループに導入すれば、予測不能なコンバータエンジンを作成できる。これによってコンバータエンジンはさらに複雑になり、デッドロック状態が発生する確率も増えることが多い。

コンバータエンジンパターンの実装では、サイクル内にリミッターを入れてポジティブエンジンを制御し、エンジンがエネルギーを生産しすぎないようにすることだ。たとえば、ターンごとに変換できる燃料リソース数が限られている場合、エンジンが稼働する最大レートに上限を設ける。マキネーションダイアグラムでは、ゲートノードを使ってリソースの流れを制限できる。自動車の場合、エンジンが燃料をエネルギーに変換し、エネルギーが燃料ポンプを駆動する。燃料ポンプはこのエネルギーの一部を消費して、もっと多くの燃料をエンジンに送る。これによって、スロットルにより制限可能なポジティブフィードバックループが作成される。

例

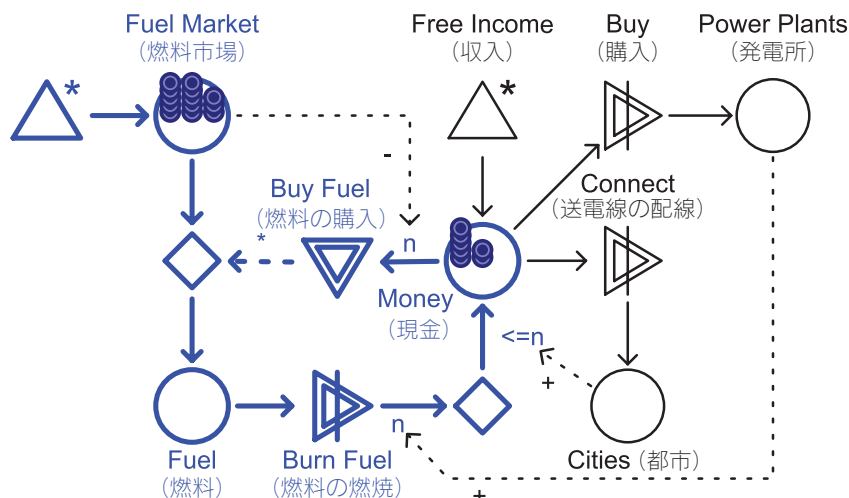
1980年代の宇宙取引コンピュータゲームEliteは、ときおりコンバータエンジンとして働く経済が特徴的だった。Eliteは、惑星ごとに独自の市場を持ち、さまざまな商品の売買取引を行うゲームである。プレイヤーはときどき、惑星Aで買った商品を惑星Bで売ってもうけを出し、惑星Aで需要の高い商品を惑星Bで調達できるような、実入りの良い取引ルートを発見する(図B.4)。取引ルートに3〜4の惑星が絡む場合もある。このような取引ルートは実質的にコンバータエンジンである。取引ルートは、プレイヤーの宇宙船の輸送容量に制限されるが、それもお金を払えば拡張可能だ。宇宙船の他の能力もこのコンバータエンジンに影響を与える可能性がある。たとえば、宇宙船の「ハイパースピード」や敵地の飛行で生き残る能力(あるいはコスト)など、すべてが取引ルートの利益に影響するのである。時間が経つと、最終的にプレイヤーの商業活動で需要が減少して価格も下がり、その取引ルートの利益は少なくなる(図B.4のダイアグラムではこのメカニズムが省略されている)。



図B.4 Eliteでの宇宙旅行と取引

プレイヤーが位置する惑星（AまたはB）によってコンバータがアクティブになり、取引メカニズムを図中央に導入する。図右側は宇宙船のアップグレード例である。

ボードゲーム「電力会社」の中核は、コンバータエンジンである（図B.5）。もっとも、コンバータの1つはもっと高度な構造に置き換えられている（第7章、174ページの「精緻化とネストのパターン」の項を参照）。プレイヤーは市場から燃料を購入するために資金を費やし、その燃料を発電所で使って資金を生産する。このゲームに設定された物語は、プレイヤーが電力を生産し販売することだ。しかし、このゲームメカニクスは電力をモデル化していない。プレイヤーが燃料を直接資金に変換しているだけだ。余ったお金は効率の高い発電所や、多くの都市に送電線を張りめぐらすことに投資される。このコンバータエンジンには限りがある。プレイヤーは送電線のつながった都市でしかお金を稼げないため、これがターンごとのお金の出力に事実上の制限をかけている。電力会社は弱いスタティックエンジンでデッドロックを回避している。プレイヤーは、発電所でお金を稼げなくても、ターンごとに少額のお金を収集できるのだ。プレイヤーは燃料を蓄積して価格を上げられ、また同時にそれが停止メカニズムとしても振る舞うため、電力会社のコンバータエンジンは若干予測不能になる。



図B.5 電力会社の生産メカニズム。青がコンバータエンジン。

関連するパターン

- コンバータエンジンはエンジン構築パターンと組み合わせると相性が良い。これは、2つのコンバータの変換レート、リミッターの設定など、エンジンの各種の設定変更のチャンスを増やせるからだ。
- コンバータエンジンが構築するポジティブフィードバックは、何らかの摩擦を導入してバランスをとるのがベストである。
- コンバータエンジンはスタティックエンジンパターンを精緻化する。
- コンバータエンジンは、エンジン構築パターンやワーカー配置パターンによって精緻化できる。

B.4

エンジン構築

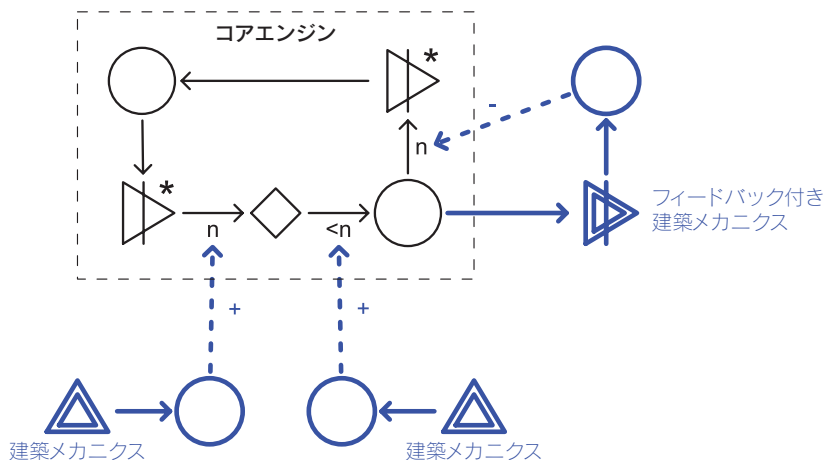
- 種類: エンジン
- 意図: ゲームプレイのかなりの部分で、プレイヤーにリソースの安定的なフローを生成するためのエンジンの構築とチューニングに専念させるようにする。
- 動機: ダイナミックエンジン、コンバータエンジン、または異なるエンジンの組み合わせによって複雑で動的なゲームコアが形成される。ゲームにはエンジンを向上させるメカニクスが最低でも1つ、可能な限り複数含まれている。このメカニクスには複数のステップを含めることができる。エンジン構築パターンによって面白いゲームプレイを生むには、プレイヤーがエンジンの状態を簡単に把握できるようではいけない。

適用可能性

エンジン構築の使用場面：

- 建設や構築にフォーカスしたゲームを作成したい。
- 長期的な戦略や計画にフォーカスしたゲームを作成したい。

構造



パーティシパント

- 一般に、**コアエンジン**は各種のエンジンを組み合わせた複雑な構造である。
- 最低でも1つ、だが通常は複数の**構築メカニクス**によって、コアエンジンを向上させる。
- **エネルギー**はコアエンジンが生産する主なリソースである。



メモ

コアエンジンのこの構造は1つの例である。エンジンには決まった構築方法はない。エンジン構築の要件は、いくつかの構築メカニクスがエンジンに作用し、エンジンがエネルギーを生産することのみである。

B

コラボレーション

構築メカニクスはエンジンの出力を増大させる。構築メカニクスのアクティブ化にエネルギーが必要な場合、ポジティブな建設的フィードバックループが作成される。

結果

エンジン構築によってゲームの難易度が上昇する。計画や戦略的な決断が必要なため、ペースのゆっくりしたゲームに最適である。

実装

何らかのかたちで予測不能性を含めることは、難易度を上げ、変化に富んだゲームプレイを作り出し、絶対優位の戦略を回避する良い方法である。コアエンジンは多くのメカニクスで構成されるため、エンジン構築によって予測不能にできる確率は高い。コアエンジン自体の複雑度によっても、ある程度予測不能性が引き起こされる。

エンジン構築パターンを使う際、ポジティブな建設的フィードバックループが強すぎず、速すぎないようにすることが重要だ。一般に、エンジン構築のプロセスをゲーム全体に拡散するとよい。

構築メカニズムをアクティブ化するのにエネルギーが必要ない場合、エンジン構築パターンはフィードバックなしで機能する。これは、エンジンが生産するエネルギーの種類によってゲームに異なる影響があり、エネルギーごとに異なる戦略をプレイヤーが使える場合に、極めて重要な構造になり得る。しかし、構築メカニズムのアクティブ化は通常何らかの方法で制限する必要がある。

ダイナミックエンジンパターンのアップグレードメカニズムもまた、構築メカニズムの一例である。実際、ダイナミックエンジンはエンジン構築パターンのシンプルで一般的な実装である。とはいえ、このシンプルさはダイナミックエンジンが1種類、あるいはせいぜい2種類のアップグレードしか許していないことを意味する。エンジン構築パターンに従うゲームの一般的なコアエンジンは、もっと多くのアップグレードの選択肢を許容する。

例

エンジン構築の好例はシムシティである。シムシティのエネルギーは資金であり、資金は大半の構築メカニズムをアクティブ化にするのに使用できる。このメカニズムは敷地の準備、区画の割り当て、インフラ構築、特殊な建物の建設、取り壊して構成される。シムシティのコアエンジンは、住民、求人、電力、輸送容量と3つの異なる種類の土地区画など多くの内部リソースがあることで、極めて複雑である。エンジン内部のフィードバックループは、プレイヤーの不注意でエンジン管理をしくじって、エンジンが崩壊するまで、あらゆる種類の摩擦を生じさせながら、主たるフィードバックループのバランスを効果的に調整し続ける。

ボードゲームのPuerto Ricoでは、各プレイヤーが新世界の植民地を構築する。植民地はさまざまな種類のリソースを生産し、このリソースを再投資したり勝利点に変換したりできる。コアエンジンにはプランテーション、建物、入植者、資金、さまざまな作物といった多くの要素やリソースが含まれる。Puerto Ricoは、エンジンを向上させるさまざまなアクションを行う有限個の地位を競い合うマルチプレイヤーゲームである。つまり、さまざまな構築メカニクスを獲得するために競い合うのだ。こうして強力なマルチプレイヤーダイナミクスが生じ、ゲームプレイの大半に貢献している。

関連するパターン

- エンジン構築パターンの難易度を上げるには、複数フィードバックを構築メカニクスに適用することが最適である。
- 構築メカニズムのアクティブ化にエネルギーを消費するタイプのエンジン構築を実装して作られた一般的なポジティブフィードバックのバランスをとるには、あらゆる種類の摩擦パターンが適切である。
- ダイナミックエンジンは、エンジン構築パターンの、最もシンプルに実装可能なものの1つだ。
- エンジン構築パターンは、ダイナミックエンジンおよびコンバータエンジンパターンを精緻化する。
- エンジン構築パターンは、ワーカー配置パターンで精緻化できる。

B.5

静摩擦

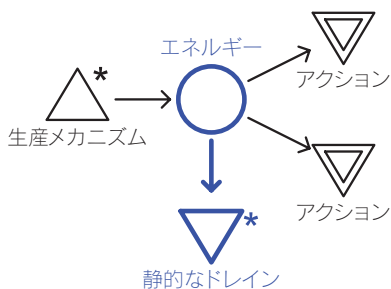
- **種類**：摩擦
- **意図**：ドレインとして、プレイヤーが生産したリソースを自動的に消費する。
- **動機**：静摩擦パターンは、定期的にリソースを消費して生産メカニズムを抑制する。その消費速度は、定数またはランダム性で設定できる。

適用可能性

静摩擦の使用場面：

- リソース生産を抑えておき、最終的にはプレイヤーが克服可能なメカニズムを作りたい。
- ダイナミックエンジン用のアップグレードに投資することで得られる長期の利益を大きくしたい。

構造



パーティシパント

- リソース：**エネルギー**
- エネルギーを消費する**静的なドレイン**
- エネルギーを生産する**生産メカニズム**
- エネルギーを消費する他の**アクション**

コラボレーション

生産メカニズムは、プレイヤーがアクションを行うのに必要なエネルギーを生産する。静的なドレインは、プレイヤーが直接制御できないところで、エネルギーを消費する。

結果

静摩擦パターンは、エンジンパターンが作ったポジティブフィードバックの逆の効果を出す比較的シンプルな手法だ。とはいえ、ダイナミックエンジンの初期の出力を削減するのに、アップグレードには何の影響も与えないため、ダイナミックエンジンにつきものの長期的戦略を増長する傾向がある。

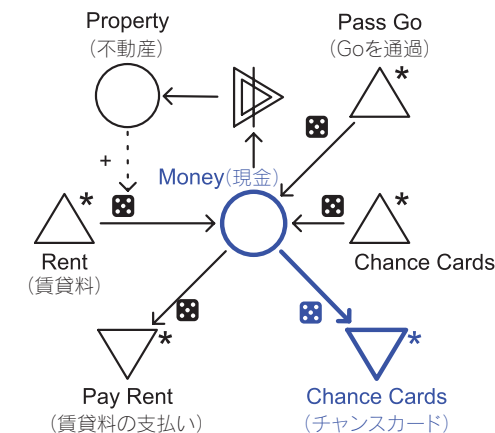
実装

静摩擦の実装において検討すべき重要事項は、消費速度を一定にするか何らかのランダム性に任せるかである。一定の静摩擦は理解することが簡単で予測可能な最たるものであるが、ランダムな静摩擦はゲームの動的な振る舞いに多くのノイズを引き起こす可能性がある。後者の生産メカニズムにランダム性を使う方法は、優れた代替手段である。摩擦をかける頻度も検討する必要がある。短いインターバルでフィードバックが適用されると、システムに断続的な振る舞いを引き起こす可能性のある長いインターバルや不規則なインターバルのときに比べると、システム全体は安定する。一般に、システムの動的な振る舞いでエネルギーが継続的に失われる場合、同じエネルギー量が周期的に失われるよりもその影響は小さい。

例

ローマ時代の都市建設ゲーム、シーザーIIIでは、プレイヤーは、ミッションごとの特定の場面で、皇帝に敬意を表する行動をとらなければならない。ミッションごとの敬意を表するタイミングは一定であり、プレイヤーの行動に影響されない。これは実質上、ごく稀に起こる高度な形式の静摩擦であり、ゲームの内部経済を大きく揺さぶるほどの影響を与えている。このゲームについての詳細な解説は第9章「経済の構築」を参照してほしい。

モノポリーのダイナミックエンジンは、静摩擦をはじめとしたさまざまな種類の摩擦によって効果を逆転させている(図B.6)。静摩擦の実装に使われる主なメカニズムはチャンスカードであり、これによってプレイヤーは稀に所持金を失うことになる。プレイヤーの保有する不動産に関連するチャンスカードもあるが、その大半は無関係だ。



図B.6 モノポリーの静摩擦

他のプレイヤーに賃貸料を支払うことも静摩擦だと考える読者もいることだろう。なにしろ賃貸料の支払い頻度やその金額は、支払う側が直接制御できないからだ。だが、賃貸料の支払いはアトリションパターンの例であって、静摩擦ではない。摩擦レートは時間とともに変化せず、プレイヤーが間接的に影響を与えられる。つまり、あるプレイヤーが優勢な場合、他のプレイヤーが劣勢である確率は高く、これによって摩擦に負の作用が働くのだ。図B.6の図は個別のプレイヤー視点であるため、この側面は含まれていない。

関連するパターン

- 静摩擦は長期的な投資メリットを誇張する。そこで、スタティックエンジン、コンバータエンジン、またはエンジン構築のパターンと組み合わせるのが最適である。
- 静摩擦は動摩擦またはスローサイクルパターンによって精緻化できる。

B.6

動摩擦

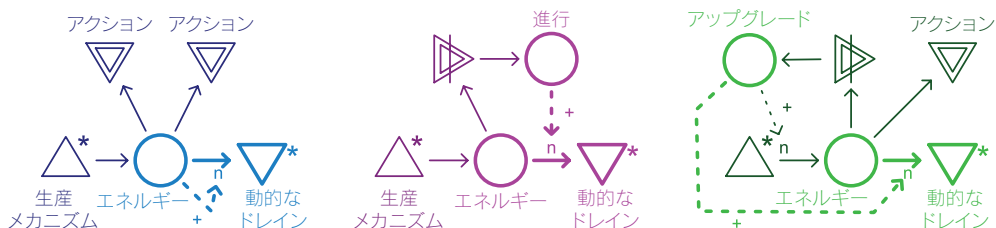
- **種類**：摩擦
- **意図**：ドレインとして、プレイヤーが生産したリソースを自動的に消費する。消費速度はゲームの他の要素の状態によって影響を受ける。
- **動機**：動摩擦は生産の効果を打ち消すが、プレイヤーのパフォーマンスに順応する。動摩擦はゲームにネガティブフィードバックを適用する伝統的な手法だ。

適用可能性

動摩擦の使用場面：

- リソースの生産が速すぎるゲームでバランスをとりたい。
- 生産の逆の効果を生み出し、自動的にプレイヤーの進行度やパワーに対応するメカニズムを作りたい。
- 短期的戦略を優先するため、ダイナミックエンジンによって作られる長期的な戦略の効果を削減したい。

構造



パーティシパント

- リソース：**エネルギー**
- エネルギーを消費する**動的なドレイン**
- エネルギーを生産する**生産メカニズム**
- エネルギーを消費する他の**アクション**

コラボレーション

生産メカニズムは、プレイヤーがアクションを行うのに必要なエネルギーを生産する。動的なドレインは、プレイヤーが直接制御できないエネルギーを消費するが、最低1つのゲームシステム内の他の要素の状態によって影響を受ける。

結果

動摩擦パターンは、エンジンパターンが作ったポジティブフィードバックの逆の効果を出す優れた手法だ。動摩擦はゲームシステムにネガティブフィードバックループを追加する。

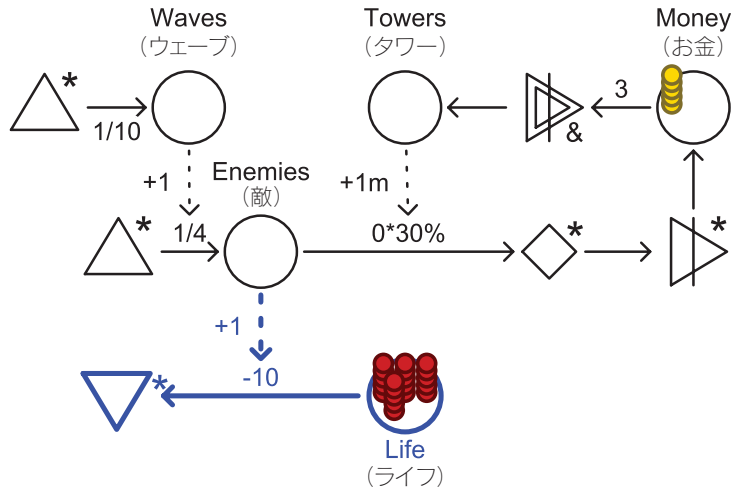
実装

動摩擦を実装する方法はいくつかある。実装の際には、消費速度^{レート}の変更を引き起こす要素の選択が重要である。一般に、この要素には、利用可能なエネルギーの量、ダイナミックエンジンかコンバータエンジンへのアップグレード数、プレイヤーの目標に対する進行状況が使われる。利用可能なエネルギー量が摩擦レートを変更する場合、ネガティブフィードバックは加速する傾向がある。プレイヤーの進行状況または生産パワーによって摩擦レートを変化させる場合、フィードバックはより間接的になり、おそらく遅くもなる。

動摩擦によってポジティブフィードバックループの効果を抑制する際、動摩擦の実装によるポジティブフィードバックループとネガティブフィードバックループの特徴の違いを考慮する必要がある。この特徴が同じ場合（双方とも等しく速く、持続性が等しいなど）、差が大きいときよりも、その効果はずっと安定する。たとえば、低速で持続性のある動摩擦が、当初は優れたリターンをもたらす高速だが持続性のないポジティブフィードバックに対して作用する場合、プレイヤーは当初は著しく進展するが、長期的には苦しむ可能性がある。高速なポジティブフィードバックと低速なネガティブフィードバックが最も頻繁に見られる組み合わせになるだろう。

例

タワーディフェンスのメカニクスは、敵が引き起こすライフポイントの動的なドレインを中心にめぐっており、プレイヤーはタワーを構築してこのドレインを制御下に置く必要がある（図B.7）。つまり、動摩擦が効果を発揮しないようにすることが、ゲームのゴールである。実際のゲームで正しい種類のタワーを配置するには、この図では省略した戦略が必要である。



図B.7 タワーディフェンスの動摩擦

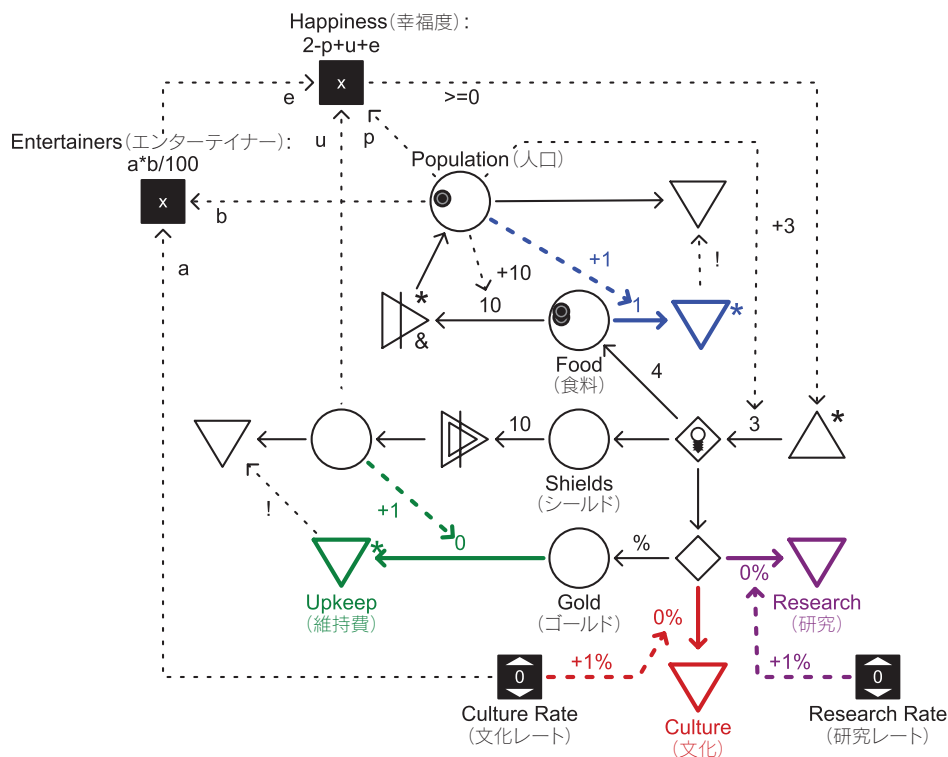
動摩擦は、シヴィライゼーションで都市の生産メカニズムに使われている (図B.8)。このゲームではプレイヤーが都市を建設することで、食料、ハンマー^{†1}、商業^{†2}を生み出す。都市が成長するにつれ、その人口に応じた、より多くの食料が必要になる。プレイヤーは食料と他のリソースの生産量をある程度制御できるが、プレイヤーに与えられた選択肢は地形によって制限されたものだ。初期段階で食料の生産量を増やすと、その間他のリソースの生産量は少なくなるが、大きな潜在力によって急速に成長する。とはいえ、急速な成長は問題を引き起こす。人口の半分以上が「幸福」でないと、市民の暴動によって生産が停止するからだ。都市の幸福度は当初2である。特別な建築物を建設するか、商業を文化に変換すれば、幸福度を上げられる。この2つの方法はどちらも、生産プロセス上において、異なるプロファイルで動摩擦を多く引き起こすものだ。特別な建築物の建設には時間も費用もかかるが、極めて持続性があり、リターン率も比較的高い。商業を文化に変換するのは高速だが、必要な投資に対するリターン率は比較的低い。

関連するパターン

- 動摩擦は、ポジティブフィードバックを引き起こし、しばしば複数フィードバックパターンの一部となるあらゆる種類のパターンのバランスをとる優れた手法だ。
- アトリションパターンは、マルチプレイヤーのインタラクションの結果である動摩擦を精緻化する。
- 動摩擦は停止メカニズムによって精緻化される。

†1 訳注：建物の構築に利用するリソースの一種

†2 訳注：お金とほぼ同義



図B.8 シヴィライゼーションの都市の経済。動摩擦メカニズムは色付きで示した。プレイヤーは文化や研究の設定を自由に調整して市民の不満や研究成果をコントロールできる。こうした設定はグローバルなものであり、すべての都市に等しく影響を与える。

B.7

停止メカニズム

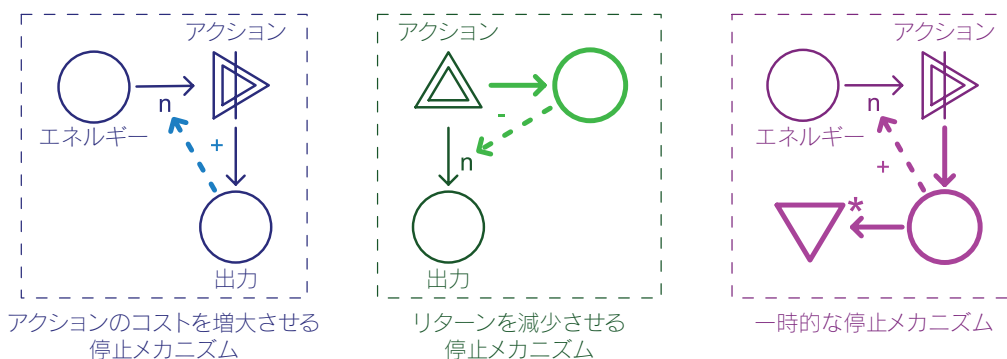
- **種類**：摩擦
- **意図**：アクティブ化されるごとに、ある特定のメカニズムの効果を削減する。
- **別名**：収穫逡減の法則
- **動機**：プレイヤーによる強力なメカニズムの乱用を防ぐため、あるメカニズムの効率を使用するごとに削減する。停止メカニズムが永続する場合もあるが、通常は永続しない。

適用可能性

停止メカニズムの使用場面：

- 特定のアクションをプレイヤーに乱用させないようにしたい。
- 絶対優位の戦略を抑制したい。
- ポジティブフィードバックメカニズムの効果を削減したい。

構造



パーティシパント

- 何らかの出力を生産する可能性のある **アクション**
- このアクションに必要なリソース **エネルギー**
- エネルギーコストを上げる、またはアクションの出力を下げる **停止メカニズム**

コラボレーション

停止メカニズムを機能させるには、アクションにエネルギーコストがあるか、リソースを生産するか、またはその両方が必要である。停止メカニズムはアクティブになるごとに、エネルギーコストを上げたりリソースの出力を削減したりして、あるアクションメカニズムの効率を削減する。

結果

停止メカニズムを使うと、ポジティブフィードバックループの影響を大幅に削減し、リターン不足にさえる。

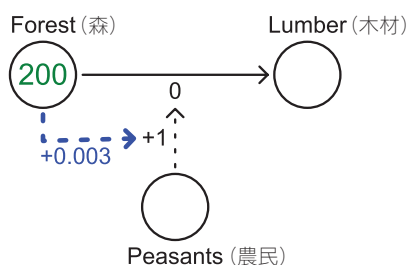
実装

停止メカニズムを実装する際には、効果を永続させるかどうかを検討することが重要だ。蓄積された出力を用いて、停止メカニズムの強さを調整する場合、その効果は永続しない。このような場合、プレイヤーは、出力を生産しては他のアクションで出力を使用することを頻繁に繰り返す必要がある。

停止メカニズムはプレイヤーごとに適用したり、複数のプレイヤーに均等に適用したりできる。後者の場合、他のプレイヤーより前に、このメカニズムの効果が弱いうちにこのアクションを使用したプレイヤーは、相対的にリワードを得ることになる。これは停止メカニズムによって、勝っているプレイヤーと負けているプレイヤーのどちらが先に行動するかに基づいたある種のフィードバックが作成できることを意味する。

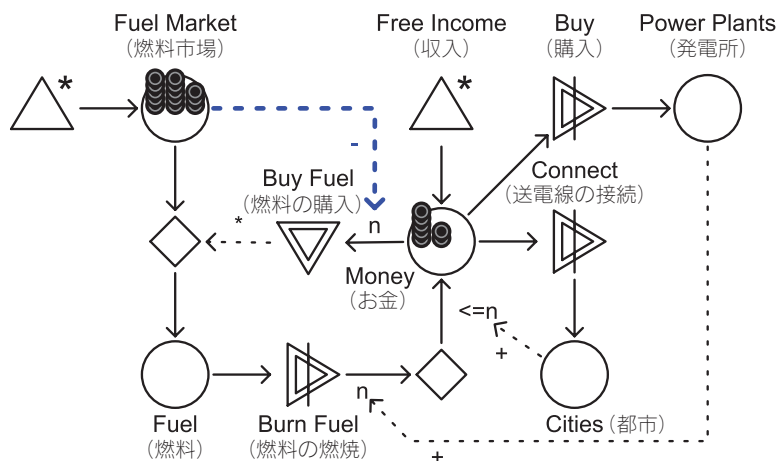
例

巧妙な停止メカニズムはウォークラフトIIIの木材収集メカニズムに見られる。ウォークラフトIIIで、プレイヤーは農民に木を切らせて木材を生産できる。農民は森からプレイヤーの基地まで木材を輸送する必要があり、輸送中は木を切れないため、森までの距離が生産メカニズムの効率に影響を与える。木を伐採すればするほど森の開拓面積が増えてこの距離は長くなる。図B.9にこれらのメカニクスを示す。



図B.9 ウォークラフトⅢの停止メカニズム。
各農民の生産速度は、森がほぼ開拓されるころには0.4まで下がる。

ボードゲーム「電力会社」の燃料市場の価格メカニズムには停止メカニズムがかかっている(図B.10)。電力会社では、プレイヤーは資金を使って燃料を買い、燃料を燃やして資金を生産する。このポジティブフィードバックループは、燃料を大量に購入すると全プレイヤーに対してその価格が上がるという事実によって効力が阻害される。電力会社ではトッププレイヤーが最後に行動するため、この停止メカニズムはトッププレイヤーに強力なネガティブフィードバックを引き起こす。



図B.10 電力会社の停止メカニズムは燃料価格をつり上げてネガティブフィードバックを引き起こす。これは特にトップのプレイヤーに影響が大きい。

関連するパターン

- 停止メカニズムは、複数フィードバックを実装したシステムでよく見られる。
- 停止メカニズムは動摩擦パターンを精緻化する。
- 停止メカニズムはスローサイクルパターンによって精緻化できる場合がある。

B.8

アトリション (消耗)

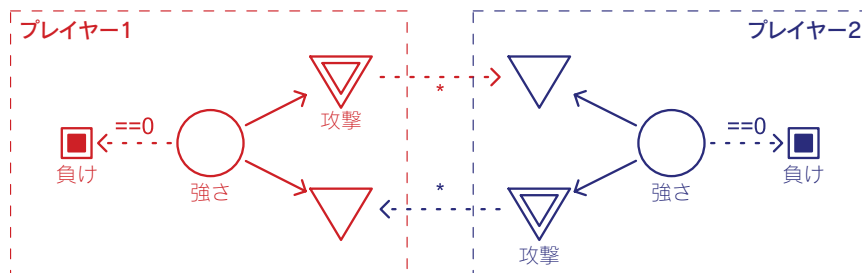
- **種類**：摩擦
- **意図**：プレイヤーが、ゲームの他のアクションに必要なリソースを、他のプレイヤーから積極的に盗んだり破壊したりできるようにする。
- **動機**：プレイヤーに互いのリソースを盗んだり破壊したりできるようにすることで、プレイヤーは優位に立つためにお互いに争って相手を排除できるようになる。

適用可能性

アトリションの使用場面：

- 複数のプレイヤーの間で、直接かつ戦略的なインタラクションを許容したい。
- 戦略的な嗜好やプレイヤーの思いつきで決まる性質のあるシステムに、フィードバックを導入したい。

構造



パーティシパント

- 同じ (または似た) メカニクスまたは選択肢を持つ複数の **プレイヤー**。
- **強さ** というリソース。強さをすべて失ったプレイヤーはゲームから脱落する。
- 他のプレイヤーの「強さ」を消滅させたり、盗んだりする特別な **攻撃** アクション。

コラボレーション

攻撃アクションを行うことで、プレイヤーたちは互いの強さをドレインできる。攻撃アクションによって、それを行ったものが強さを失う場合も失わない場合もある。攻撃の実行と引き替えに強さを失わない場合は、攻撃アクションに時間がかかるようにするか、またはスキルやランダム性を成否に組み込むなどの何らかの手段が必要になる。攻撃のコストとその効力、ゲームの他のアクションにとっての利益のバランスによって、攻撃の効果とアトリションパターンの優位性が決まる。

結果

アトリションによって、他のプレイヤーを破壊できる手段を直接コントロールできるため、システムに多くのダイナミズムが生まれる。あるプレイヤーの現在の状態によって他のプレイヤーの反応が引き起こされるため、破壊的なフィードバックが頻繁に導入される。勝利条件の性質やゲームの現在の状態にもよるが、このフィードバックは、プレイヤーたちが共謀してトッププレイヤーに反撃する場合はネガティブになり、より弱いプレイヤーを攻撃して排除するような場合はポジティブになる。



メモ

建設的フィードバックと破壊的フィードバックは、ポジティブフィードバックとネガティブフィードバックと同じではないことを思い出してほしい。これらの違いについては第6章、132ページの「フィードバックの7つの特徴」の項を参照のこと。

実装

アトリションパターンをうまく機能させるには、他のことにも使えるリソースをあえて攻撃に投資させる必要がある。この投資をしないとした場合、2人用ゲームのアトリションは、戦略的な選択を伴わない、単に相手を破壊する競争になってしまう。プレイヤー間の社会的インタラクションを促すマルチプレイヤーゲームでは、攻撃する相手を選ぶ必要があるため、投資なしの攻撃でもある程度うまくいく。

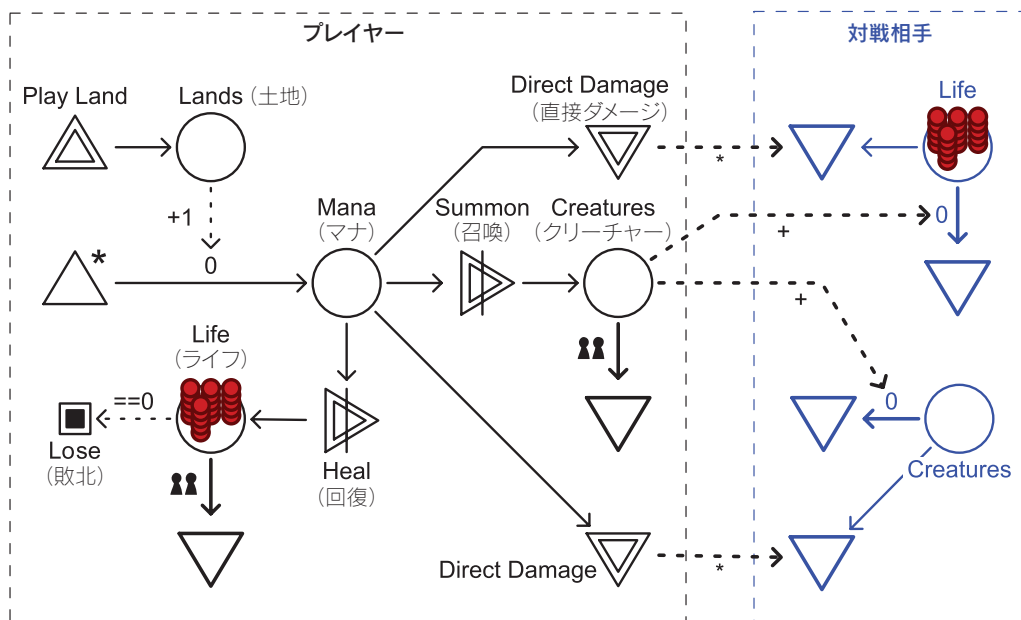
アトリションの実装には、「強さ」という1つのリソースだけでなく、ライフとエネルギーなど、2つのリソースを使うのが最も一般的である。プレイヤーはエネルギーを使ってアクションを行い、ライフがなくなるとゲームで負けになる。2つのリソースを使う際には、ある程度の関連があることが重要だ。プレイヤーはエネルギーを使ってライフを獲得できることがよくあるが、ライフとエネルギーの関連が明確でない場合もある。たとえば、エネルギーを獲得するか、ライフを獲得するかのをどちらかを選択する必要がある場合、一般に両者を同時に行えないため、この2つのリソースには暗黙的なリンクがある。

2人用ゲームのアトリションでは、ゲームに他のアクションも用意される必要があり、3人以上のプレイヤー向けゲームではプレイヤーが別のアクションを行えることが多い。たいいていの場合、これらのアクションは、プレイヤーの攻撃や防衛の効力を高める強さを生産する何らかのメカニズムを構成する（そしてアトリションパターンを軍拡競争パターンに精緻化する）。大半のリアルタイムストラテジーゲームにはこれらの選択肢がすべてそろっており、選択肢ごとに複数のバリエーションがあるゲームも多い。

勝利条件と相手プレイヤーを排除する効果は、アトリションパターンに大きなインパクトを与える。とはいえ、勝利条件が相手の排除である必要はない。プレイヤーはアトリションパターン外でポイントを稼いだり、特定の動機に達したりする可能性があり、それによって利用可能な戦略の数が自動的に増える。他のプレイヤーを攻撃したり排除したりするとボーナスがもらえる場合、アトリションパターンはより弱いプレイヤーの排除をプレイヤーに促すことができる。

例

トレーディングカードゲームのマジック：ザ・ギャザリングは、高度なアトリションパターンを実装している。この実装を示したのが図B.11だ。ただし、1人のプレイヤーのみの視点で詳細が表現されている。



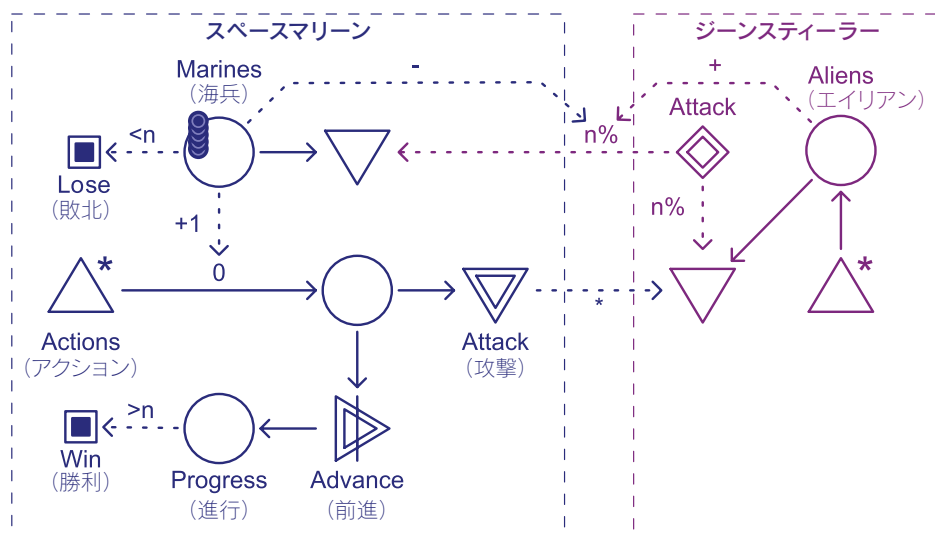
図B.11 マジック：ザ・ギャザリングの消耗戦メカニズム

マジック：ザ・ギャザリングでは、ターンごとにプレイヤーは1枚のカードを使用できる。このカードによってプレイヤーは土地を追加したり、クリーチャーを呼び出したり、回復のための呪文を唱えたり、敵や敵のクリーチャーに直接的なダメージを与えたりできる。だが、土地以外のアクションを行うには、すべて魔法エネルギー「マナ」を使う必要がある。マナをたくさん持てば持つほど、ターンごとに使える量が増えて、もっと強力なアクションを行える。クリーチャーは他のクリーチャーと戦い、自分のクリーチャーすべてを失うと、敵から直接的なダメージを受ける。ライフポイントをすべて失ったプレイヤーはゲームから脱落する。マジック：ザ・ギャザリングは、ライフとエネルギー（マナ）に別のリソースを使ってアトリションを実装したゲームの例である。

マジック：ザ・ギャザリングでは、ゲームプレイの選択肢によって異なるアトリションが展開される。直接的なダメージは簡潔にドレインをトリガーする。ドレインはその名のとおりで、即効的で直接的だ。その反対に、クリーチャーの呼び出しは敵のクリーチャーとライフに対して永続的なドレインをアクティブにする。この効果は直接的なダメージほど強力ではないものの、時間の経過とともに蓄積されるため、大きな破壊力を持つ可能性がある。使える選択肢とその強

度はプレイヤーが持つカードによって決まる。プレイヤーは自分の膨大なコレクションでデッキを作るため、マジック：ザ・ギャザリングではデッキの作成が重要である。

アトリションの実装で最も明白な手法はゲームを対称的にすることだ。しかし、多くのシングルプレイヤーゲーム、そしてある種のマルチプレイヤーゲームでさえ、非対称のアトリションを使っている。非対称のアトリションを使った例としてボードゲームのSpace Hulkが挙げられる。あるプレイヤーはわずかな宇宙海兵隊員「スペースマリーン」をコントロールしてミッションを達成しようとし、もう一方のプレイヤーは、無限に供給されるエイリアンの「ジーンスティラー」をコントロールしてこのミッションを阻止しようとする。ジーンスティラーのプレイヤーはスペースマリーンの数削減してミッションの達成を阻止しようとし、ジーンスティラーがスペースマリーンを十分に破壊すると勝利を収める。スペースマリーンのプレイヤーはジーンスティラーを破壊しても勝てないが、ジーンスティラーの数を低く抑えておかないと生き残れない。これは、ジーンスティラーの数が増えると破壊力が増すからだ。Space Hulkのこのメカニクスの意図を示したのが図B.12だ。



図B.12 Space Hulkの非対称の消耗戦パターン

関連するパターン

- アトリションパターンはあらゆる種類のエンジンパターンとともに使用できる。破壊的でなく建設的で、ほぼ常にネガティブなマルチプレイヤーフィードバックの代替形式としてはトレードが使える。
- アトリションは動摩擦パターンを精緻化する。
- アトリションパターンは、軍拡競争パターンやワーカー配置パターンで精緻化できる。

B.9

チャレンジのエスカレーション

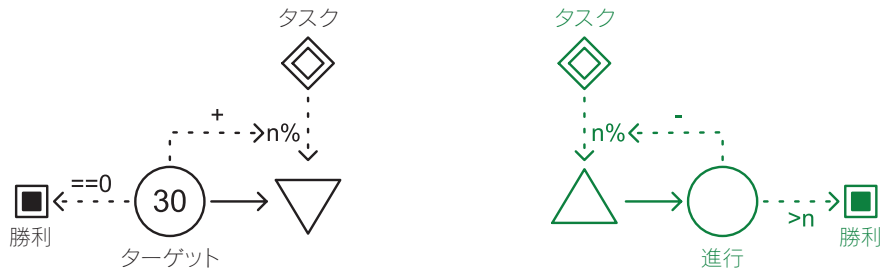
- **種類**：エスカレーション
- **意図**：ゴールに近づけば近づくほどに進行を難しくする。
- **動機**：プレイヤー進行とゲーム難易度間のポジティブフィードバックループによって、プレイヤーがゴールに近づくほど難易度がどんどん上がる。これにより、プレイヤーのスキルレベルにゲームが瞬時に順応できる。優れたパフォーマンスによってプレイヤーの進行がぐっと早まる場合は特に重要となる。

適用可能性

チャレンジのエスカレーションの使用場面：

- プレイヤーのスキル（通常は身体的なスキル）に応じたテンポの速いゲームで、プレイヤーが前進するごとに難易度を上げたい場合。つまり、プレイヤーが進行するほどタスクを完成する能力が得られにくくする。
- あらかじめデザインしたレベル進行を（部分的に）置き換える創発性メカニクスを作成したい。

構造



パーティシパント

- 未解決のタスクを表す**ターゲット**。
- プレイヤーのゴールへの進行状況を表す**進行**。
- ターゲット数の削減または進行の生成を行う**タスク**。
- プレイヤーがゴールに近づくか、またはターゲット数を削減するにつれゲームの難易度を上げる**フィードバックメカニズム**。

コラボレーション

タスクはターゲット数の削減、進行の生成、またはその両方を行う。フィードバックメカニズムによって、プレイヤーがゴールに近づくほどタスクの難易度が上がる。

結果

チャレンジのエスカレーションは、シンプルなポジティブフィードバックループに基づいてゲームの難易度に影響を与える。このメカニズムによってゲームの難易度はプレイヤーのスキルレベルに合うように素早く調整される。タスクの失敗がゲームの終了につながる場合、チャレンジのエスカレーションはゲームを非常に速いものにする。

実装

チャレンジのエスカレーションパターンを実装するタスクは、一般にプレイヤーのスキルによって影響を受ける。チャレンジのエスカレーションパターンがゲームのコアメカニクスの大半を占めている場合は特にそうだ。タスクがランダムメカニクスまたは決定性メカニクスの場合、プレイヤーはゲームの進行をまったく制御できない。チャレンジのエスカレーションパターンはもっと複雑なゲームシステムの一部となり、プレイヤーが成功するチャンスに何らかの間接的な制御ができる場合のみ、ランダムメカニクスまたは決定性メカニクスが効力を発揮する。マルチプレイヤーダイナミクスを使うのも選択肢の1つだが、もっと複雑なゲームシステムのほうがおそらく好ましいだろう。

例

チャレンジのエスカレーションパターンの典型的な例はスペースインベーダーである。スペースインベーダーでは、侵攻するエイリアンをすべて破壊して、画面最下段に到達されないようにしなければならない。プレイヤーがエイリアンを破壊するごとに、他のエイリアンはスピードを少し上げるため、より撃墜しにくくなる。

もう1つの例はパックマンだ。パックマンのタスクは、レベル内のすべてのエサを食べることだが、モンスターを追いかけると、最後のエサにたどり着くのがどんどん難しくなる（パックマンの詳細と図については第5章「マキネーション」を参照）。

関連するパターン

チャレンジのエスカレーションに静摩擦または動摩擦を組み合わせると、プレイヤーの能力に素早く難易度を一致させるゲームが構築できる。

B.10

複雑度のエスカレーション

- **種類:** エスカレーション
- **意図:** プレイヤーがゲームを制御し抵抗をしようとし続ける限り、ポジティブフィードバックによって複雑度を増大させて、ついにはプレイヤーを打ち負かす。

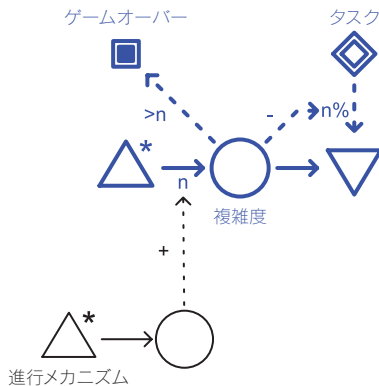
- **動機**：失敗すると複雑化し、難易度が上がるアクションをプレイヤーにタスクとして与える。プレイヤーがついて行ける限りプレイを継続できるが、ポジティブフィードバックが暴走しだすとゲームはすぐに停止する。ゲームが進行するにつれ、複雑度を作り出すメカニズムもスピードアップし、ある地点に達すると、プレイヤーはついて行けなくなって確実に負けるようになる。

適用可能性

複雑度のエスカレーションの使用場面：

- プレッシャーの高いスキルベースのゲームを目標にする。
- あらかじめデザインしたレベル進行を（部分的に）置き換える創発性メカニクスを作成したい。

構造



パーティシパント

- ゲームが生成する**複雑度**を、プレイヤーがある限界以下に制御する必要がある。
- プレイヤーによって複雑度が削減可能な**タスク**。
- 時間とともに複雑度が増す**進行メカニズム**。

コラボレーション

複雑度はただちにさらなる複雑度の増大を促し、制御する必要のある強力なポジティブフィードバックループを生み出す。複雑度が自分の能力を超えて制御できなくなるとプレイヤーは負ける。

結果

十分なスキルがあれば、プレイヤーは複雑度の増加に長い間持ちこたえることができるが、いったんついて行けなくなると、複雑度は急激に上昇してゲームがただちに終了する。

実装

複雑度のエスカレーションパターンを実装するタスクは、一般にプレイヤーのスキルによって影響を受ける。複雑度のエスカレーションパターンがゲームのコアメカニクスの大半を占めている場合は特にそうだ。タスクがランダムメカニクスまたは決定性メカニクスで制御されている場合、プレイヤーはゲームの進行をまったく制御できない。ランダムメカニクスまたは決

定性メカニクスは、プレイヤーが成功するチャンスのある程度制御できるより複雑なゲームシステムに置かれれば、ある程度うまく機能する。マルチプレイヤータスクを使うのも選択肢の1つだが、より複雑なゲームシステムのほうがおそらく好ましいだろう。

複雑度の生成をランダムにすると、ピーク時はついて行くのがやっとなる可能性もあるが、複雑度が若干緩んだときにほっと息をつけるような、変化に富んだペースのゲームが作成できる。

進行メカニクスを実装する方法は、複雑度の生成をシンプルなタイムベースにするもの（構造に例示したもの）から、プレイヤー本人や他のプレイヤーのアクションに依存した複雑な構成までいろいろある。このように、タスクの実行結果としてポジティブフィードバックを進行メカニズムに導入すると、複雑度のエスカレーションにチャレンジのエスカレーションを組み合わせることが可能だ。

複雑度のエスカレーションからもたらされる複雑度を、複数の別個のフィードバックループに入力すると、複数フィードバック構造の一部としてより良く機能する。たとえば、複雑度の生成を制御するかなり低速なネガティブフィードバックループにタスクを入力して、複雑度のエスカレーションを部分的にバランスさせることができる。

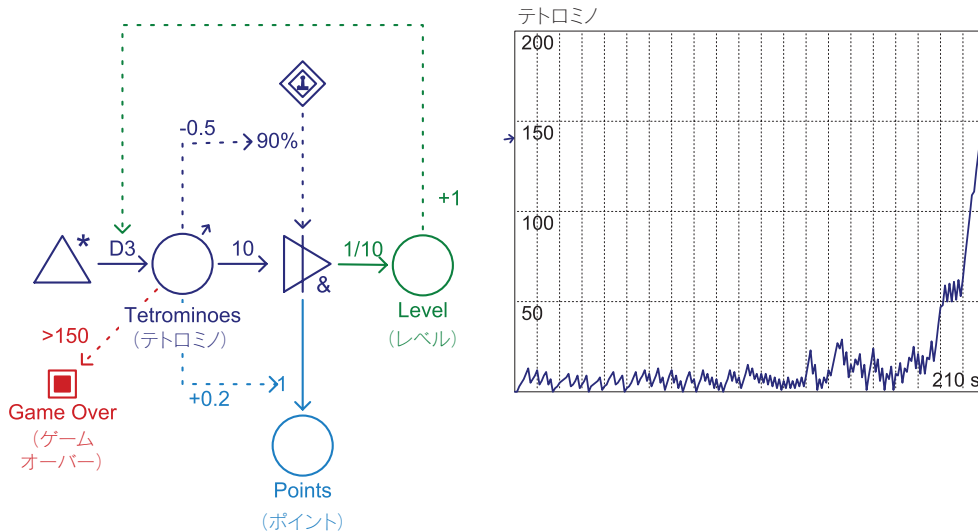
例

テトリスでは、テトロミノが一定の速度で落下することで複雑性が生成される。この複雑性の生成では、時間の経過とともに形の異なるテトロミノが作成されるため、若干のランダム性がある。プレイヤーはテトロミノがぴったり合うように配置する必要がある。1つの段が完全に埋まると段ごと消滅し、新たなテトロミノを落とす空間が広がる。プレイヤーがペースの維持に遅れるとテトロミノは急速に積まれるため、テトロミノを配置する時間がどんどん少なくなる。このような仕組みにより、プレイヤーが注意を怠ると、フィールドの複雑度はどんどん増大し、テトロミノが画面最上段に達するとプレイヤーは負けてしまう。テトリスでは、レベルによって進行メカニズムが作られる。プレイヤーが10段クリアするごとに次のレベルに上がってテトロミノの落下速度が速くなり、正確に配置することがどんどん難しくなる。つまり、このレベルメカニズムはチャレンジのエスカレーションパターンの例でもある。

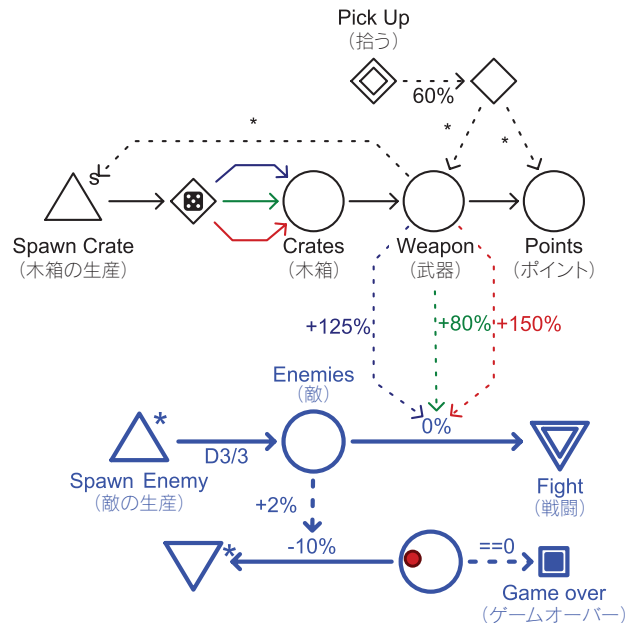
テトリスのこれらのメカニズムを図B.13に表した。この図ではテトロミノをポイントに変換した。フィールド内のテトロミノの数が増えるとポイント数が上がる。これによって、一度に複数の段をクリアし、ハイリスク・ハイリターン戦略の可能性を表現した。図B.13のグラフは、プレイヤーが追いつけないほどペースが上がると、ゲームが急激に制御不能になることを示している。

アクションシューターのインディーズゲームSuper Crate Boxでは、プレイヤーはさまざまな武器の入った木箱を拾い上げながら、敵を撃ち落とし続ける。敵に触れるとプレイヤーは即座に死亡する。敵は画面の最上段に出現し、まっすぐ下りてきて最下段にくと消滅する。最下段に到達した敵は最上段から再び出現するが、二度目はずっと高速で移動する。プレイヤーが使える武器は一度に1つで、必ずしもすべてが強力な武器とは限らない。勝つための唯一の方法は木箱

を拾って武器を交換することだ。プレイヤーは拾い上げた武器を有効活用するしかない。より高い得点をあげるために、プレイヤーは敵を殺してその数を制御可能な数まで減らすことと、木箱を拾うこととを交互に繰り返す必要がある。図B.14にSuper Crate Boxのダイアグラムを示す。



図B.13 テトリスのエスカレーションする複雑度



図B.14 Super Crate Boxでは、プレイヤーはポイント獲得と敵の数を制御下に置くことを交互に行う必要がある。

関連するパターン

- 進行メカニズムの実装にはあらゆる種類のエンジンパターンが使える。
- 進行メカニズムをチャレンジのエスカレーションパターンとして実装することが一般的だ。

B.11

軍拡競争 (アームズレース)

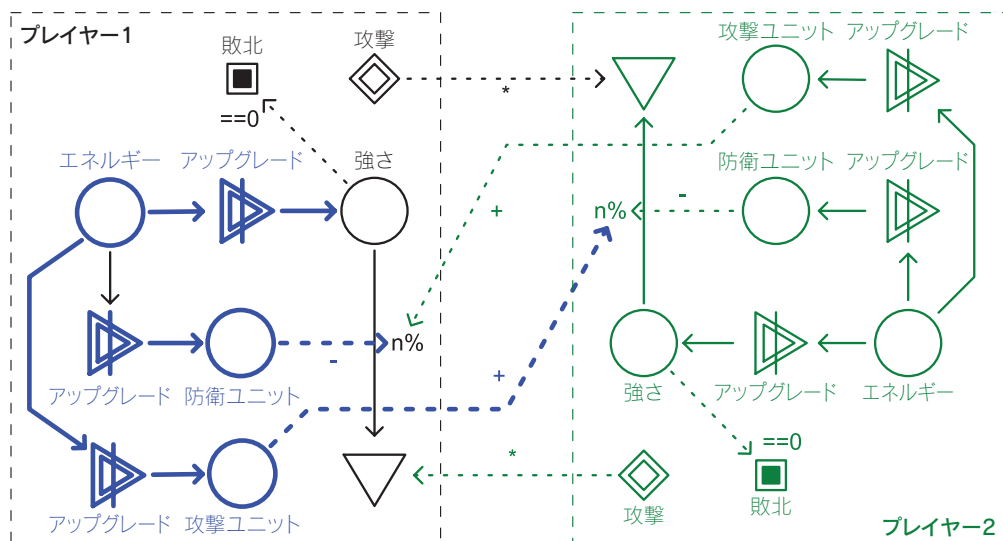
- **種類**: エスカレーション
- **意図**: プレイヤーが他のプレイヤーに対抗するために、攻撃および防衛の軍事力にリソースを投資することができるようにする。
- **動機**: 攻撃および防衛の軍事力に投資させることで、ゲームに多くの戦略的な選択肢を導入する。プレイヤーは自分のスキルや好みに合った戦略を選択できる。

適用可能性

軍拡競争の使用場面:

- さまざまな戦略的な選択肢を与えたい。また、アトリションパターンを使った絶対優位の戦略を回避したい。
- ゲームのプレイ時間を長くしたい。
- スキルや好みに合った戦略やプレイスタイルを築くことをプレイヤーに促したい。

構造



パーティシパント

- 同じ（または類似の）攻撃メカニズムをアクティブ化できる複数の**プレイヤー**。
- **強さ**のリソース。「強さ」をすべて失ったプレイヤーはゲームから脱落する。
- アップグレードによって消費される任意選択の**エネルギー**リソース。エネルギーと「強さ」が同じものである場合がある。
- 最低1つの**アップグレードメカニズム**によって各プレイヤーの攻撃または防衛の軍事力が向上する。

コラボレーション

攻撃メカニズムによってプレイヤーたちは互いの強さをドレインしたり盗んだりできる。攻撃やアップグレードメカニズムをアクティブにするために、プレイヤーはエネルギーまたは時間を投資する必要がある。アップグレードメカニズムによって、攻撃または防衛の軍事力を向上させたり、プレイヤーの強さを回復させたりできる。

結果

軍拡競争によって、プレイヤーには、いろいろと試すことができる戦略的選択肢が多く導入されて、ゲームのバランスをとるのが難しくなる可能性がある。一般に、アップグレードの選択肢にはジャンケンのような3すくみのメカニズムを実装し、どの戦略にもそれを打ち破る戦略が存在するようにするのがベストである。たとえば、中世を題材にした多くの戦争ゲームでは、重装歩兵は騎兵隊を破り、騎兵隊は大砲を破り、大砲は重装歩兵を破る。つまり、最善の戦略にして最も効果的な軍隊構成は、部分的に、対戦相手の選択によって決まるところがある。

戦略的な選択肢を豊富にすることでプレイヤーは自分のスタイルや戦略を構築できる。たとえば、気に入ったメカニズムを何度も使い、気に入らないメカニズムは無視してもかまわない。軍拡競争パターンを使うと、当初は防衛的にプレイをするという選択肢があるため、一般にゲームは長くなる。対立や紛争を長時間に延ばすことさえできる。

実装

軍拡競争を実装する際は、アップグレードに投資できるリソースをどれに決定するかが重要である。強さとエネルギーが同一のものの場合、投資をしすぎると、自分を弱くする可能性がある。これはアップグレードに時間がかかる場合は特にそうだ。エネルギーと強さを別のものにした場合、エネルギーと強さの関係を慎重に検討する必要がある。強さによってエネルギーの生産速度を決定してもよいだろう。これによって強力でポジティブな破壊的フィードバックループができあがる。エネルギーを強さに変換させたり、エネルギーを投資して時間とともに強さを生産させたりすることも可能だ。選択肢は実にさまざまである。

軍拡競争によってゲームが長くなりすぎるのを回避するには、プレイヤー全員に同じリソースを収集させるように仕向けるか、アップグレードには強さを投資する必要があるようにする

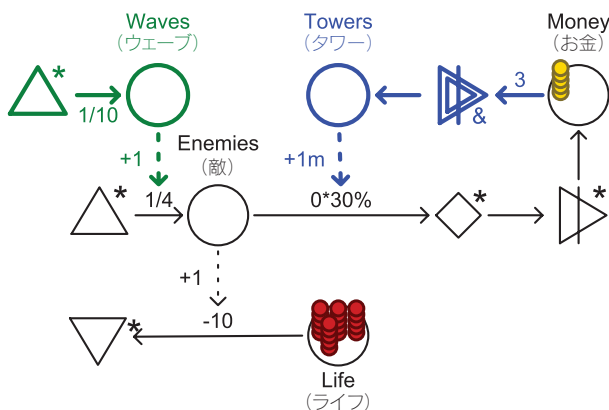
など、アップグレードをアクティブ化するリソースに対して競争を激しくするのがベストである。

軍拡競争は対称である必要はなく、非対称な軍拡競争も可能ではあるが、バランスをとるのはずっと難しくなる。

例

多くのリアルタイムストラテジーゲームが軍拡競争パターンを実装している。たとえば、StarCraft IIやウォークラフトIIIでは、プレイヤーがユニットの軍事能力を向上させるためのテクノロジーを検討できる。こうしたゲームでは、強さはプレイヤーのユニットと建物の合計になり、エネルギーは労働者ユニットによって収集され、アップグレードや新しいユニットの構築に使われる。

軍拡競争はタワーディフェンスゲームでもよく見られるが、これは、このパターンを非対照的に実装したものだ。たとえば、図B.15の青と緑のメカニズムは、青いプレイヤーと敵（緑）の攻撃軍事力を増大する2つの異なるメカニズムを表している。大半のタワーディフェンスゲームではもっと多くのアップグレードメカニズムが使われている。プレイヤーはタワーをアップグレードしたり、異なる効果のタワーを選んだりできる一方で、敵の突撃波（ウェーブ）には、プレイヤーが別種の反応をする必要がある別種の敵を含められる。



図B.15 タワーディフェンスゲームの非対称な軍拡競争

関連するパターン

- 軍拡競争パターンはダイナミックエンジンと組み合わせてエネルギーと強さを生産できる。この組み合わせは多くのリアルタイムストラテジーゲームに見られる。
- 軍拡競争はアトリションパターンを精緻化する。
- 軍拡競争パターンはワーカー配置パターンで精緻化できる。

B.12

プレイスティルの強化

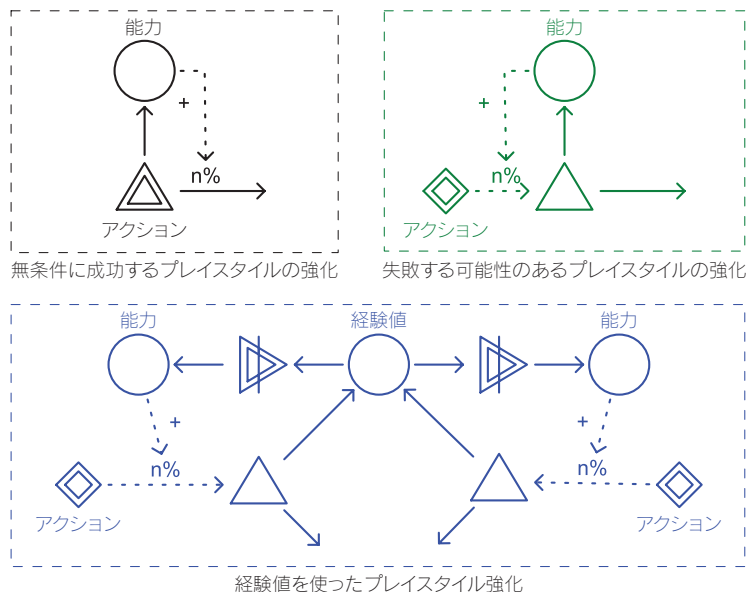
- **種類**：その他
- **意図**：プレイヤーのアクションに低速でポジティブな建設的フィードバックをかけて個性化を奨励し、プレイヤーの好みのプレイスタイルにゲームをじわじわと順応させる。
- **別名**：ロールプレイングゲーム (RPG) 要素
- **動機**：プレイヤーのアクション（ゲームに別の効果を与えるアクション）への低速でポジティブな建設的フィードバックによって、時間とともにプレイヤーのアバターまたはユニットを成長させる。アクションそのものがこのメカニズムにフィードバックされるため、アバターまたはユニットは時間とともに個性化し、特定のタスクをうまく実行できるようになる。複数の有効な戦略と個性化がある限り、アバターまたはユニットは時間とともにプレイヤーの好みやスタイルを反映するようになる。

適用可能性

プレイスタイルの強化の使用場面：

- 複数のセッションにまたがるゲームで、プレイヤーに長期の投資をさせたい。
- プレイヤーが構築、事前の計画および個人的な戦略の立案をすることに対するリワードを提供したい。
- プレイヤーに特定の戦略に没頭したり、ある役割になりきったりしてもらいたい。

構造



パーティシパント

- プレイヤーが行う**アクション**。その成功度はプレイヤーキャラクターまたはそのアクションにかかわるユニットの属性にある程度依存する。
- **能力**（アビリティ）というリソース。アクションが成功する確率に影響を与え、時間とともに成長する。
- **経験値**という任意選択のリソース。能力を増大させるのに使用できる。ゲームによっては「スキルポイント」と呼び、交換不能な経験値とは別のリソースにする場合もある。

コラボレーション

- 能力はアクションの成功率に影響を及ぼす。
- アクションを試みると経験値が生産されるか、または直接的に能力が向上する。アクションが成功する必要があるゲームもあれば、必要のないゲームもある。
- 経験値は能力の向上に費やせる場合もある。

結果

プレイスタイルの強化は、複数のセッションにわたって長時間プレイするゲームで最も効果を発揮する。

プレイスタイルの強化は、ゲームにおいて実際に有効な選択肢として複数の戦略とプレイスタイルが存在しないとうまく機能しない。選択肢が1つしかないか、またはあっても数個程度の場合、すべてのプレイヤーが同じ戦略を使う結果になって、ゲームが面白くなる。

プレイスタイルの強化によって、プレイヤーにミニマックス法の振る舞いを奨励する可能性がある。これは、強力なアバターやユニットをできる限り速く獲得するために、とり得る最善の選択肢を求める戦略のことだ。ミニマックス法が成功した場合、通常それは絶対優位の戦略になる。フィードバックの強さがすべてのアクションと戦略に均等に行きわたらない場合にこの事態が発生する。

プレイスタイルの強化を導入すると、経験のあるプレイヤーが未経験者より有利になる。経験を積んだプレイヤーは、自分の選択肢や自分のアクションによる長期的な影響をよく理解しているからだ。

プレイスタイルの強化によって、ゲームを遊ぶことに最も時間を費やすプレイヤーにリワードが与えられる。つまり、ゲームに費やした時間がプレイヤー間のスキルレベルの格差というかたちで返ってくるわけだが、これが期待される副作用の場合もあれば、**そうでない場合もある**。プレイスタイルの強化のゲームで、時間が経ってから戦略を変えるのは良い結果をもたらさない可能性がある。プレイスタイルを変えると、それまでに投資したメリットを失うからだ。

実装

プレイスタイルの強化を実装する際、経験値を使うかどうかの決定が重要になる。経験値を

使うと、(アバターの)成長とアクションに直接的な結び付きがなくなり、ある戦略で収集した経験値を、別の戦略で活躍するスキル構築に使うことが可能になる。逆に、経験値を使わないと、アクションの頻度に対するフィードバックのバランスを確実にとる必要がある。つまり、頻繁に行われるアクションは、頻度の低いアクションより弱いフィードバックにする必要がある。

ロールプレイングゲームはプレイスไตล์の強化パターンを中心に構築されたゲームの典型である。こうしたゲームでは、一般にフィードバックループは極めて低速にして、チャレンジのエスカレーション、動摩擦、または停止メカニズムでバランスをとって、アバターが急激に成長しないようにする必要がある。実際、こうしたゲームの大半は、初期は進行が早く、しだいに遅くなるようにバランスがとられているが、これは、必要な経験値の投資が急激に増加するからだ。

また、フィードバックを生成するのにアクションの成功を要件としかどうかを決定する必要がある。この決定によってプレイヤーの振る舞いは劇的に変化する。成功が要件となる場合、フィードバックループは影響を受ける。つまり、タスクの難易度がアクションの成功にも影響するようにして、プレイヤーにさまざまな難易度のタスクで試練を与えてアバターのトレーニングができるようにするのがベストであろう。経験ポイントを得るのに成功が要件ではない場合、プレイヤーはゲーム後半の困難な段階で、それまで無視していた能力を向上させるチャンスが増える。とはいえ、どんな場面でも特定のアクションばかり行おうとするプレイヤーもあり、そのアクションにリスクがない場合は特に意図しない非現実的で滑稽な結果になる可能性がある。

例

ペンと紙を使ったロールプレイングゲームの多くがプレイスไตล์の強化パターンを導入している。たとえば、ウォーハンマーRPGやVampire: The Masqueradeでは、プレイヤーがゴールに到達すると経験値がもらえる。この経験値は自分のキャラクターの能力を向上させるのに使える。奇妙にも、ロールプレイングゲームの元祖であるダンジョンズ&ドラゴンズにはプレイスไตล์の強化がない。ダンジョンズ&ドラゴンズでは、次のレベルに行くためにためておく必要がある経験値がプレイヤーに与えられる。しかし、次のレベルに行っても、プレイヤーは自分のキャラクターの能力を向上させる上で何の影響も与えられない。つまり、キャラクター能力はプレイヤーのプレイスไตล์や好みに順応しないのである。

コンピュータのロールプレイングゲームThe Elder Scrolls IV: Oblivionでは、アバターの成長がそのアクションと直接結び付いている。アバターの能力は、関連するアクションを行った回数にそのまま比例しているのだ。The Elder Scrolls IV: Oblivionは、プレイスไตล์の強化を経験値なしで実装している。

シヴィライゼーションIIIにはゲームで勝つためのさまざまな方法が用意されている。プレイヤーは自分が選択した軍隊、経済、文化、科学(または任意の組み合わせ)の戦略を優位にするために、自分の戦略を優先させる都市発展や「七不思議」と呼ばれる特殊な建造物を構築するこ

とができる。シヴィライゼーションIIIでは、経験値の役割を果たすリソースがいくつかある。その代表的な例が資金と生産だ。資金や生産のリソースがゲームの特定の戦略に結び付けられているとは限らない。ある都市で生産された資金は、別の都市の生産性の向上に使うことが可能である。

関連するパターン

プレイスタイルの強化パターンがアクションの成功に依存する場合、強力なフィードバックが生じる。この場合、停止メカニズムを使って、能力の新しいアップグレードにかかるコストを増大させることだ。

B.13

複数フィードバック

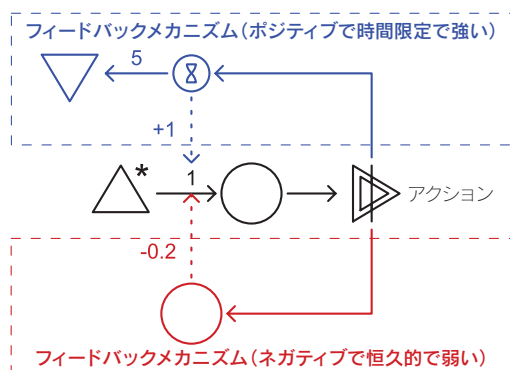
- **種類:** その他
- **意図:** 単一のゲームプレイのメカニズムが、それぞれ異なる特性を持つ複数のフィードバックメカニズムに送られる。
- **動機:** プレイヤーのアクションが一度に複数のフィードバックループをアクティブ化する。一部のフィードバックループは、他のフィードバックより明確になることがあるだろう。これによって、短期的にはアクションの結果や成果を正確に予測可能である一方で、長い目で見れば予測不能な結果が生じるかもしれないという状況を作ることができる。

適用可能性

複数フィードバックの使用場面:

- ゲームの難易度を上げたい。
- 現在のゲーム状態を読み取るプレイヤーの能力に重点を置きたい。

構造



パーティシパント

- プレイヤーがアクティブ化できる**アクション**
- アクションによってアクティブ化された複数のフィードバックメカニズム



構造のサンプルには、2つのフィードバックメカニズムがある。アクション（黒）は、ポジティブで時間限定で強いフィードバックメカニズム（赤）をアクティブ化するが、同時にネガティブで恒久的で弱いフィードバックメカニズム（青）もアクティブ化する。これは、複数フィードバックループを準備するための1つの方法をただ示したにすぎない。方法は他にもたくさんある。

コラボレーション

アクションは、複数のフィードバックメカニズムをアクティブ化し、最終的にはそのアクション自身にフィードバックされる。

結果

プレイヤーにとって、複数フィードバックループを理解するのは、単一のフィードバックループに比べて難しい。結果的に、このパターンを使用することにより、ゲームの難易度は上がることになる。

もしアクションがアクティブ化するフィードバックループが、ゲームプレイ中に（頻繁に）変わる動的な特性を持っていたとしたら、プレイヤーにとって今現在の特性を読み取ることができるかは大変重要である。なぜならば、それら特性のバランスがゲーム中に大幅に変化する可能性があるからである。

複数フィードバックループの間に正しいバランスを見つけることは、このパターンを使用するゲームにおいての重要な課題である。

実装

複数フィードバックを使ってゲームを作成する場合、それぞれのフィードバックループの特性に違いを持たせることがとても重要になる。特に、本パターンを効果的に働かせようとするときは、フィードバックの速度は変化に富むようにすべきである。あるいは、時間をかけてフィードバックの特性を変えていくとうまく機能させることができる。この目的の達成のために、1つ以上のフィードバックループにプレイスタイルの強化や停止メカニズムを加えることは、デザイン上の良い戦略となる。

複数フィードバックの最も一般的な組み合わせは、高速かつ建設的なポジティブフィードバックと、低速なネガティブフィードバックである。これにより短期的な利益と長期的な不利益との間のトレードオフが生み出される。

例

シムシティの経済は多くの複数フィードバックメカニズムを含んでいる。たとえば、都市はエネルギーを必要とするため、プレイヤーは発電所を建設する必要がある。短期的には、発電所は、住宅、商業、工業地域に活力を与え、経済的発展に拍車をかけるだろう。しかし、長期的には、発電所は汚染の原因となり、周囲地域に悪影響をもたらすことになる。同様に、道路などのインフラは、都市を成長させるために必要とされるが、長期的に見れば、使用される頻度に応じて交

通渋滞や大気汚染などといった問題を引き起こすようになる。

ボードゲームのRiskにおける攻撃は、さまざまなスピードと強さのバリエーションを3つのポジティブフィードバックループに注ぎ込む（第6章「一般的なメカニズム」のRiskの説明を参照）。明らかなことだが、軍隊を使ってより多くの土地を占拠すると、プレイヤーはより多くの軍隊を構築することができる。2つ目のフィードバックループはカードである。プレイヤーは攻撃を成功させるとカードを獲得し、特定のカード3枚の組み合わせによってさらなる軍隊の増強が可能になる。このカードは遅めのフィードバック形態を実装する。3つ目のフィードバックは、大陸を占領することで生成される。大陸はターンごとにプレイヤーに対してボーナスとして相当な数の軍隊を与えてくれることになる。これは非常に高速かつ強力なフィードバックループになるが、プレイヤーがそれを達成するためには、より高い投資が要求される。

関連するパターン

プレイスタイルの強化と停止メカニズムは、アクションによって経時変化が起こるフィードバックの特性を確保するのに良い方法である。

B.14

トレード

- **種類**: その他
- **意図**: プレイヤー同士のトレードは、マルチプレイのダイナミクスと建設的なネガティブフィードバックを導入することができる。
- **動機**: プレイヤーが重要なリソースを取引することが許される。通常、後続のプレイヤーたちがリードしているプレイヤーたちに追いつくためにお互いに助け合うことができる一方で、リードしている側にとっては厳しい交渉に直面することを意味する。リソースの流れが不安定であるか均等にプレイヤーの間に行き渡っていないとき、あるいはその両方であるとき、特にトレードはうまく機能する。

適用可能性

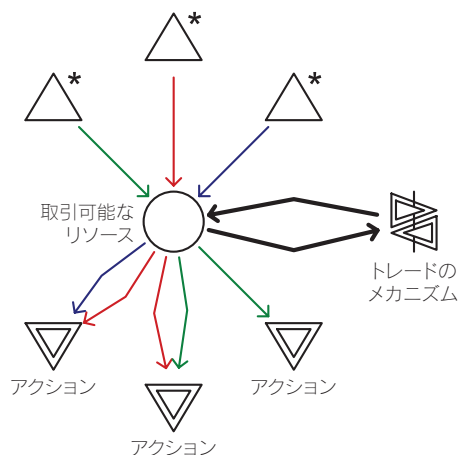
トレードの使用場面:

- ゲームにマルチプレイヤーのダイナミクスを導入したい。
- 建設的なネガティブフィードバックを導入したい。
- プレイヤー同士が（戦闘などとは対照的に）商取引を通じてインタラクションすることを奨励するソーシャルなメカニクを導入したい。

コラボレーション

取引可能なリソースは、トレードのメカニズムを使用してプレイヤー同士で交換することができる。

構造



パーティシパント

- リソースをプレイヤー間で取引可能にする
- トレードのメカニズム**
- さまざまな手段で交換や使用ができる、複数の**取引可能なリソース**
- 取引可能なリソースを使用することを要求する**アクション**

結果

トレードは、ゲームの進行を実際にスローダウンさせることはないが、通常、(破壊的ではないため) 遅れをとっているプレイヤーの追い上げを助けるような、ネガティブフィードバックを導入する。

トレードは、社会的で物々交換スキルに長けるプレイヤーに有利に働く。

実装

ボードゲームにおいては、トレードを実装するのは非常に容易である。いつどのようにプレイヤーがリソースの取引ができるのかを指定するだけでよい。マルチプレイヤーのコンピュータゲームでも、トレードの実装は容易であるが、コンピュータが制御するキャラクターが介在するトレードの仕組みを作るのは簡単ではない。

良くできたトレードのメカニズムを実装するには、複数の取引可能なリソースが必要とされ、これらのリソースの生産速度は変動しているか、少なくともプレイヤー間で生産速度が異なっていなければならない。トレードは、取引を行うもの同士のリソース配分に不均衡が生じているときにのみ機能する。取引可能なリソースを消費するアクションを多く組み入れたり、一度に複数の種類のリソースを消費するアクションを作ったりするのもよい。プレイヤーがそれぞれ異なるアクションを選択するとき、その不均衡な状態がさらに増強されるためだ。

例

「カタンの開拓者たち」では、村や街が生産するリソースが、新しい村や街を作るのに使われる。これは、プレイヤーが不確実なダイナミックエンジンを構築しているのと同じである。これらのエンジンのランダム性は、すべてのプレイヤーが現在ターンを取っているプレイヤーとの間でリソースの取引を可能にすることによって、部分的に対抗し合っている。為替レートは、相互の合意によって設定され、通常、リソースの可用性とプレイヤーの順位によって決定される。リードしているプレイヤーは、自分のリソースのために多くを支払うことができる。決着が近づいているとき、プレイヤーは取引を成立させてはいけないことに気づくかもしれない。

シヴィライゼーションIIIでは、プレイヤーは戦略的資源、資金、知識を交換することができる。この交換は互いに利益をもたらし、弱いほうの文明はかなり迅速に追いつけようになる。

関連するパターン

- アトリションは実質的には建設的でなく破壊的であるから、複数フィードバックの代替ソースに使える。

B.15

ワーカー配置

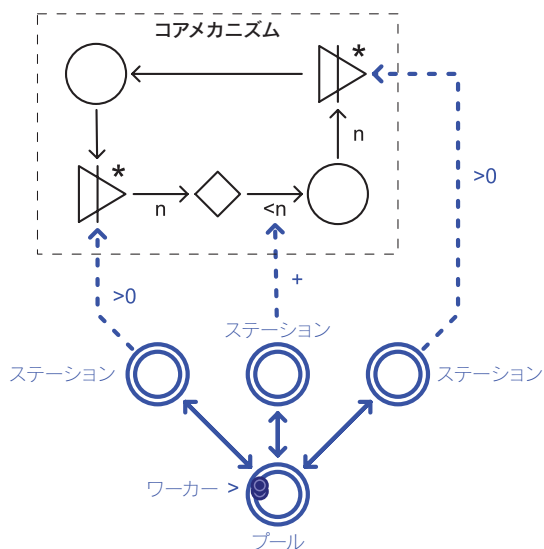
- **種類**: その他
- **意図**: プレイヤーは、ゲーム内で異なるメカニズムをアクティブにしたり改善することに専念する限られたリソース（ワーカー）を制御する。
- **動機**: 一連のメカニズムによって複雑で動的なゲームコアを作成する。プレイヤーは、一連のメカニズムをアクティブにするための限られたリソース（ワーカー）をどのように配分するかを選択しなければならない。ワーカーの数は限られているため、プレイヤーはゲームメカニズムが最も効果的になるよう操作するためにワーカーの配分を変更する必要がある。

適用可能性

ワーカー配置の使用場面:

- プレイヤーのタスクとして、継続的なマイクロマネジメントを導入したい。
- プレイヤーが環境の変化に対応することを促したい。
- 戦略の成功に重要な要素として、タイミングを組み込みたい。
- 間接的な競合のための、巧妙なメカニズムを作りたい。

構造



コンバータエンジンとしてのコアメカニズムの構造は一例である。ワーカー配置は十分に複雑なメカニズムの任意のセットに適用することができる。ワーカー配置は、ワーカー用のいくつかのステーションが特定のメカニズムをアクティブ化したり改善するために割り当てられることだけを必要とする。

パーティシパント

- **コアメカニズム** (通常は複数のメカニズムを組み合わせた複雑な構造)
- コアエンジンをアクティブ化しないし改善する複数の**ステーション**
- 異なるステーションに割り当てられる**ワーカー**のリソース
- 仕事を持ってないワーカーが集められている任意のワーカーの**プール**

コラボレーション

ワーカーはコアメカニズムをアクティブ化または改善するために異なるステーションに配置される。つまり、ワーカーはコアメカニズムを操作することになる。ワーカーは比較的容易にステーション間の移動ができるので、即座にコアメカニズムの振る舞いを変更することが可能である。

結果

ワーカー配置では、プレイヤーがワーカーのステーション間を移動するのに時間を費やす必要がある。ゲームのペースはこれを配慮する必要がある。そうしてプレイヤーは配分変更が求められるゲームイベントに備えることができるべきである。

ワーカー配置は、ワーカーが制御するコアメカニズムの振る舞いが、その時々で変更の必要に迫られる場合、最も理にかなう働きをする。これは、さまざまなゲームプレイフェーズを持つ複雑なゲームに最適である。

ワーカー配置は、通常、プレイヤーに絶えず自身のワーカーを管理することを要求するので、結果的にゲームの経済を容易に支配することができるだろう。

実装

ワーカー配置を実装するときは、ステーションの数とともにワーカーの数のバランスをとることが重要である。ワーカーの数がゲーム中に変化しないのであれば、このバランスは、ワーカーの配分の定期的な変更と固定の間に違いを生み出すことができる。

ワーカーの数が少なければ少ないほどプレイヤーは頻繁に状況に対応しなければならないが、多ければそこまで頻繁に対応する必要はない。

ワーカーの数が多い場合や、プレイヤーがゲーム内で追加のワーカーを生産できるときは、プレイヤーがすべてのステーションを操作できるようになり、もはやワーカーの配分に気を使う理由がなくなるような状況を作ってしまうよう、注意しなければならない。

これを防ぐ1つの方法は、複数のワーカーをまとめて単一のスポットに配置すると、その効果がさらに高まるようにすることである。この構造サンプルでは、中間のステーションがそのケースに当たる。

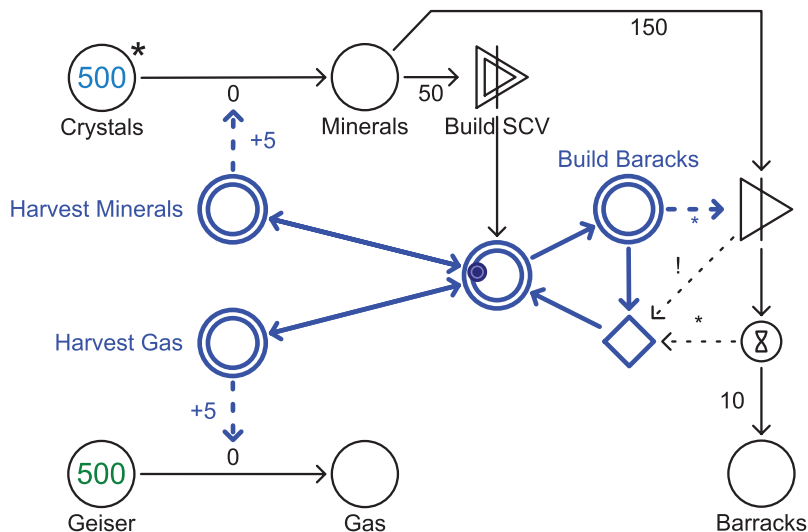
ワーカー配置を実装するゲームには、複数のプレイヤーが同じステーションで競合できるものが多い。たとえば、複数のプレイヤーが、自身の経済のために金を生産する目的で、同じ金鉱にワーカーを配置する必要があるとしよう。プレイヤーたちが同じステーションで競合しているとき、ワーカーがステーションから強制的に排除されるメカニズムを持たせることが重要になる。ターンが終わるごとに自動的にすべてのワーカーをプールへ戻すか、プレイヤーに対戦相手のワーカーを排除するもっと直接的なアクションが許されていれば、シンプルに解決できる。ステーションにおける競合は、プレイヤーが重要なステーションをブロックしてお互いの計画を邪魔し合うことにより、プレイヤー間に巧みで間接的な競争を作り出す。

ワーカー配置は、システムに動摩擦を加える多くの機会を提供する。ワーカーを配置する際にリソースを消費するか、ステーションへのワーカー配置が一定の維持費を必要とする場合、動摩擦が生み出される。どちらの場合も、より多くのワーカーを配置すると、より多くの利点を得られる反面、より多くのリソースを消費する。同時に、配置がリソースを消費する場合（このときは維持費はかからないとする）、ワーカー配置を変化させることにはペナルティが科せられる。これにより、適応可能性を下げ、計画的な振る舞いにはリワードを与えるようなワーカー配置のバージョンを作成できる。

例

StarCraftでは、さまざまなタスクを割り当てることができる宇宙用建設車両 (SCV) ユニットがワーカーである。SCVは、鉱物とガスというゲームの二大資源を採取したり、プレイヤーの拠点の建造物を建築したり、修復することができる。プレイヤーは自分が最適と思う数のSCVユニットを構築でき、同じ（または類似の）タスクに多くのSCVユニットを割り当てられることが多い。StarCraftでは、すべてのプレイヤーがマップ上の同じ場所からリソースを採取できるため、ステーションを巡り競合が起こる。これは、いくつかのレベルでは重要なフィーチャーだが、

ほとんどのレベルは、プレイヤーが一部のリソースへの比較的安全で独占的なアクセスを持った状態で始まる。図B.16は、StarCraftのワーカー配置のメカニクスを表している。

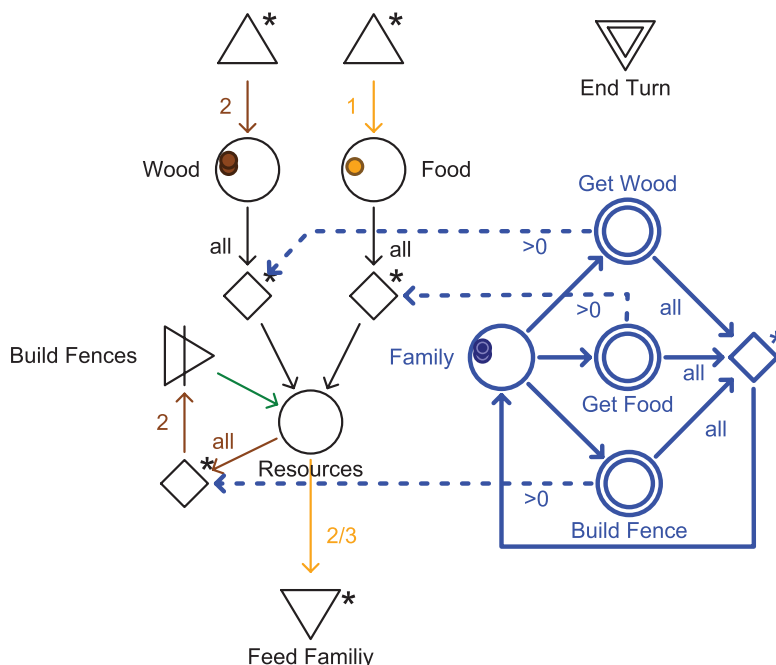


図B.16 StarCraftにおけるワーカー配置

Agricolaは、18世紀の農場を構築し運営するボードゲームである。プレイヤーは2人家族からゲームを開始する。プレイヤーの家族は、プレイヤーのワーカーである。ワーカーには、作物の種をまいたり、フェンスを立てたり、材木や他の資源を集めたりなど、さまざまな仕事を割り当てることができる。ターンごとに、プレイヤーはワーカーに新しいタスクを割り当てる必要がある。重要なタスクは、成長していく家族を養うために十分な食料を収集することである。Agricolaでは、それぞれのタスクに割り当てられるワーカーは1人でであり、プレイヤーたちは同じステーションを巡って競合することになる。リソースに関わるあるタスクを誰も実行しないターンがあった場合、そのリソースは蓄積される（たとえば、誰も木材を集めなければ、木材が溜まっていく）。このルールゆえに、タスクに関連する利点は常に変化していく。図B.17はAgricolaにおけるメカニズムをいくつか示している。

関連するパターン

- ワーカー配置は、ほぼすべての他のパターンを精緻化できるが、特にコンバータエンジン、エンジン構築、アトリション、軍拡競争（アームズレース）に有効である。
- ワーカーの数またはワーカーを配置することによって起こる動摩擦は、ワーカー配置パターンにネガティブフィードバックを適用するのに良い方法である。



図B.17 Agricolaの一部のメカニズムにおけるワーカー配置

B.16

スローサイクル

- **種類:** その他
- **意図:** 異なる状態の間をゆっくりと循環するメカニズムで、ゲームのメカニズムに周期的な変化を生み出す。
- **動機:** プレイヤーのコントロール外に、ゆっくりと動作するメカニズムを導入することで、ゲームの経済は異なるフェーズの間を緩やかにシフトする。これには、プレイヤーがより汎用性の高い戦略に適応したり、それを開発したりする必要がある。

適用可能性

スローサイクルの使用場面:

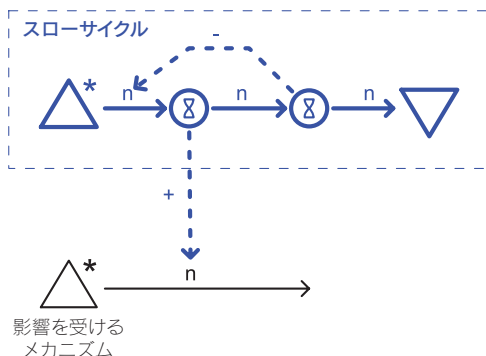
- ゲームに断続的なフェーズを導入することにより、バリエーションをさらに増やしたい。
- 絶対優位の戦略の影響力を抑制したい。
- 環境が変化し続けることに対して、プレイヤーが連続的に戦略を対応させるように強制したい。
- ゲームに精通するまでに必要な習得の期間を長くしたい（プレイヤーがスローサイクルを体験する頻度が少ないと、そこから習得する機会が減る）。

- プレイヤーがサイクルの長さや大きさを操作できるようにすることで、巧妙かつ間接的に戦略的なインタラクションを導入したい。

コラボレーション

スローサイクルの状態は、スローサイクルに影響を受けるメカニズムとインタラクションする。

構造



パーティシパント

- **スローサイクルメカニズム**は、2つ（またはおそらくそれ以上）の状態の間を揺れ動く
- **影響を受けるメカニズム**はスローサイクルの状態に依存する



ここに示した構造は一例に過ぎない。スローサイクルを構築するには多様な方法があり、そして他のゲームメカニズムに影響を与えられる数多くの手段がある。

結果

スローサイクルの影響を読み取ることは難しい場合が多く、特にゲームを始めたばかりのプレイヤーにとってその傾向が顕著だ。このことによりゲームプレイの学習曲線は長いものとなって、プレイヤーの経験の長短の違いによる優劣の差をより悪化させてしまうかもしれない。

ほとんどのケースでは、プレイヤーがスローサイクルメカニズムに影響を与えることは（仮にあったとしても）わずかである。これは、プレイヤーに明らかにサイクルの現在の状態を伝えることが重要であることを意味する。ゲーム経済に表面上ランダムに見える変化を引き起こすスローサイクルは、一般的にアンフェアなものと思なされる。

実装

スローサイクルを実装するには多くの方法がある。スローサイクルは2組の状態（たとえば、それが定期的に別のメカニズムをアクティブ化または非アクティブ化するかもしれない）の間を交互に行き来したり、もっと段階的に2つの状態の間を遷移したりするかもしれない。スローサイクルがすべてのプレイヤーに均等に影響を及ぼすことができるのが理想である。スローサイクルは、ゲーム経済のフェーズがシフトすることを予測し、それに備えるプレイヤーの能力

をテストする。ゲームの世界のコンテキストでは、スローサイクルはプレイヤーのコントロールを超えた季節の変化、潮の満ち引きや、景気循環などとして特徴づけることができる。

スローサイクルは、その周期にランダム期間を導入することによって、決定論的な傾向を抑制できる。こうすることで、プレイヤーは現在の周期の状態により注意を払うことが必要になる。周期をあまり決定論的にしないようにする別の方法は、周期の振幅をランダム化することである。たとえば、スローサイクルにより、10ターンごとの短い期間にある種のエネルギーが生成されるとする。この場合、短い期間をランダムとすることも、生成されるリソース数をランダムとすることも、周期性に影響を与えることなく実現できる。

例

第10章「レベルデザインとメカニクスの統合」の「メカニクスの中のさまざまな構造に焦点を当てる」の項(242ページ)で述べたように、StarCraft IIは、さまざまなレベルで異なるスローサイクルのメカニズムをバラエティ豊かに使用している。

ボードゲームのCaylusでは、プレイヤーは城とそれに付随する街を建造する。ゲームは3つのフェーズに分割され、各フェーズの終わりには、プレイヤーは城への貢献に対してリワードを受け取るか、あるいは貢献の欠如のために罰せられる。3つのフェーズは巧妙にスローサイクルのメカニズムを生成する。プレイヤーは自身の行う貢献を注意深く計画する必要がある。城の建造を助けるリソースを収集するワーカーを配置する上で、プレイヤー間に競合が起こるためだ(Caylusもワーカー配置パターンを実装している)。それに加えて、プレイヤーの組み合わせたアクションは、現在の周期をスピードアップしたり、スローダウンしたりするかもしれない。周期とそれが他のプレイヤーの計画にどのような影響を与えるかを予測する能力は、このゲームでは効果的かつ高度な戦略である。

関連するパターン

- スローサイクルは、スタティックエンジン、静摩擦、および停止メカニズムのパターンを精緻化する。
- スローサイクルは、ゲーム経済のフェーズをシフトさせるため、プレイヤーがその変化に的確に対応できるよう、ワーカー配置パターンと組み合わせると良い。