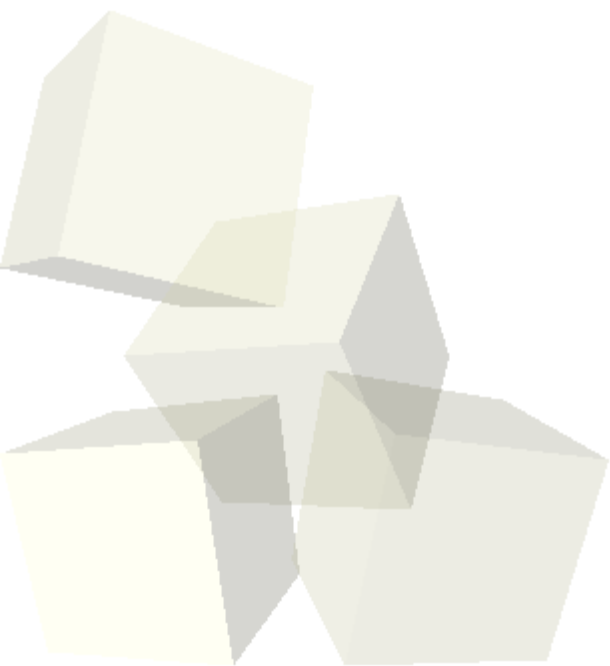




# グループ3

## 最終発表



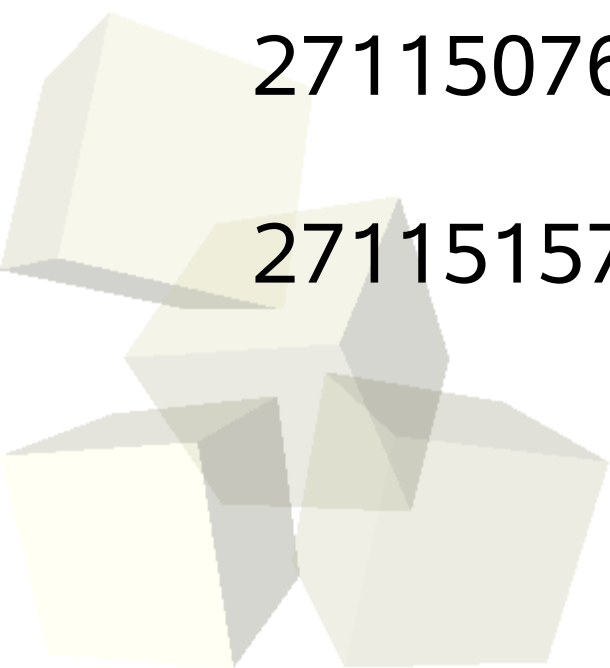


27115077 鈴木進也: 競合解消戦略の改良

27115120 丹羽貴敏: ブロック操作の可視化(3D)

27115076 鈴木祥太: 食事をする哲学者問題

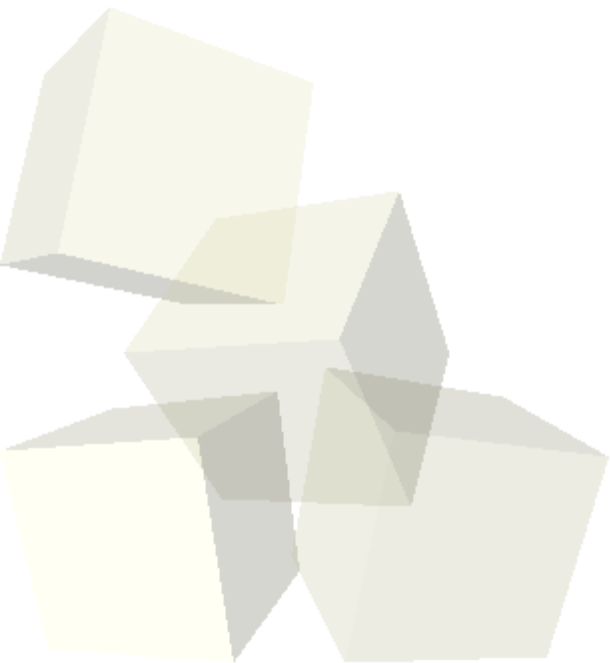
27115157 横尾由春: 食事をする哲学者問題





# 競合解消戦略の改良

## ブロック操作の可視化(3D)





## ■ 競合解消戦略としてLEXを採用

- ・現在の状態にタイムタグを付与
- ・Ifリストが最新のタイムタグを持つワーキングメモリとマッチしたオペレータを採用する

## ■ 可視化はUnityエンジンを用いて実装

- ・初期状態の入力、プランニングの実行はJavaのGUIで実装
- ・ブロックの移動する様子を3D空間で描画



- 初期状態と目標状態はそれぞれファイル読み込みによる入力
- 初期状態、目標状態の追加・削除はGUI上で可能
- プランニングで得られたプランはファイルに出力
- 得られたファイルをUnity上で読み込んで描画





## ■ LEXによる競合解消手順

- 1.すでに実行されている操作は削除
- 2.競合オペレータのIFリストと現在の状態をマッチング
- 3.最新のタイムタグを持つ状態とマッチングしたオペレータを採用
- 4.IFリストの要素が多いオペレータを採用
- 5.ランダムにオペレータを採用





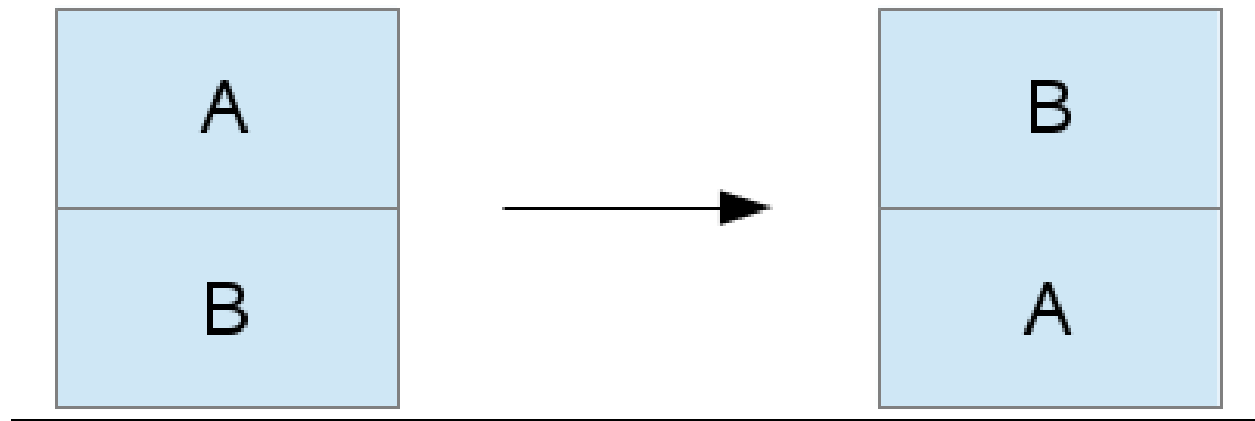
## ■ 副目標が”handEmpty”のとき・・・

- ・Place ?x on ?y が必ず採用されてしまう
- ・探索が終了しない、不適切な操作が起こるといった問題

## ■ 解決策

- ・競合する”Put ?x down on the table”と”Place ?x on ?y”の順番を交互に入れ替えるメソッドを用意





## ■ 順番を入れ替える操作を加えたとき

\*\*\*\*\* This is a plan! \*\*\*\*\*

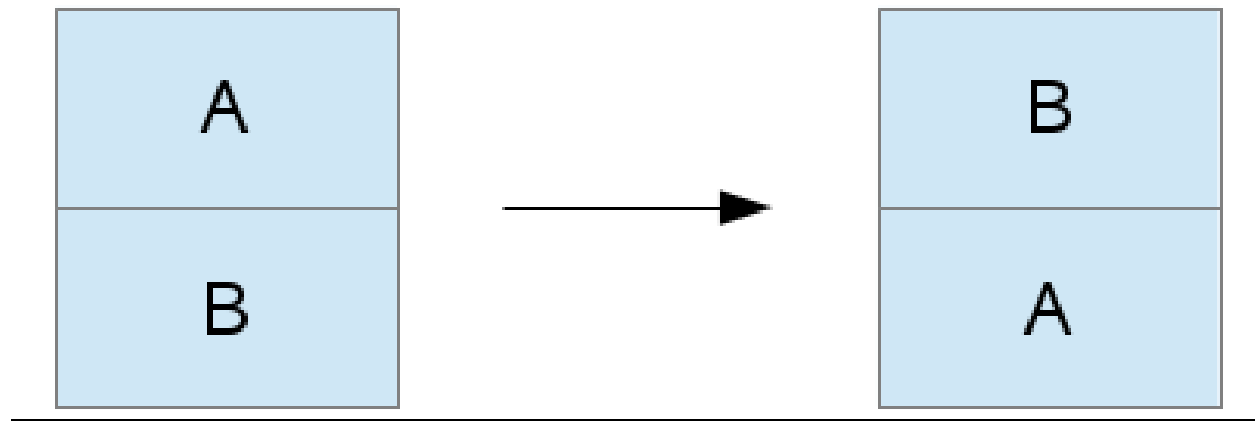
remove A from on top B

Put A down on the table

pick up B from the table

Place B on A





## ■ 単にLEXを用いたとき

\*\*\*\*\* This is a plan! \*\*\*\*\*

remove A from on top B

Place A on B ← "Put A down on the table"が採用されてほしい

pick up B from the table

Place B on A

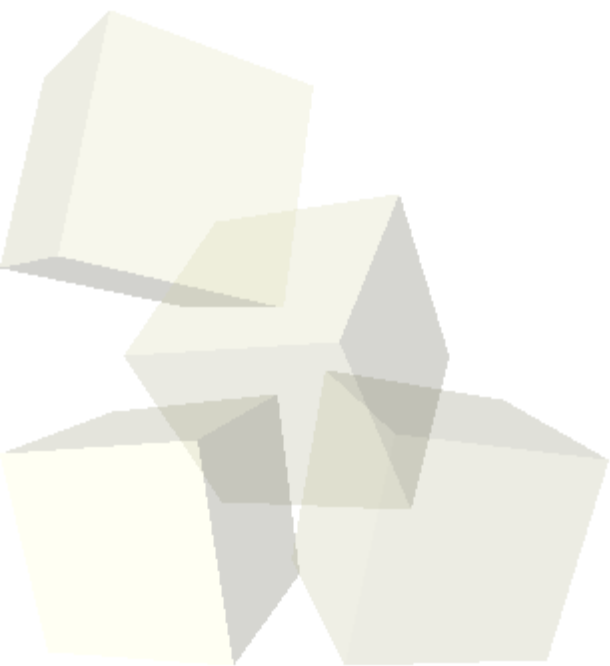


## ■ 競合解消のために指標を設けることが有効

- ・今回のタイムタグの様な各オペレータを評価する値で比較

## ■ LEXだけでは適切な競合解消は出来ない

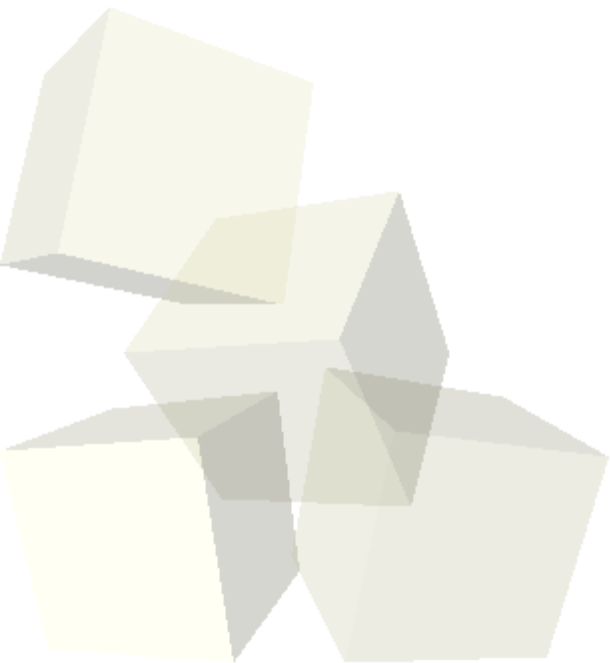
- ・問題に応じたヒューリスティックな工夫が必要





# プランニングの拡張

## ～食事をする哲学者問題～





## ■「食事をする哲学者」問題

- ・ 思索→空白→食事→思索
- ・ 空腹時、両脇の箸(またはフォーク)が使用できれば食事可能
- ・ 空腹のまま長時間待たされると餓死
- ・ すべての哲学者を死なせない方法を考える問題

プロセスが複数リソースを  
要求する場合





## ■ 食事をする哲学者問題をプランニングにより解く

- ・ 課題7のプランニングをベースにしたプログラム
- ・ 状態とオペレータを新たに定義

## ■ もちろんGUIも実装

- ・ 課題7で作成したGUIをベースに実装



## ■ 状態

- ・ 哲学者の各手の状態
  - philosopher A has nothing on the right hand :  
哲学者Aは右手に何ももっていない
  - philosopher B has fork 2 on the left hand :  
哲学者Bは左手に2番フォークを持っている
- ・ フォークの状態
  - ontable fork 1 :  
1番のフォークがテーブルに置かれている
- ・ 哲学者の食事の状態
  - philosopher C is eating :  
哲学者Cは食事をしている





## ■ オペレータ

- ・ フォークを取る動作  
pick up fork 1 from the table for philosopher A 's left hand :  
哲学者Aはテーブルからフォーク1を取る
- ・ フォークを置く動作  
put fork 3 down on the table for philosopher D 's right hand :  
哲学者Dはテーブルにフォーク3を置く
- ・ 食事を始める動作  
Eating philosopher C :  
哲学者Cは食事を始める





## ■ オペレータを動的生成

- ・ 初期状態から哲学者の人数をカウントし、その人数に応じた数のオペレータを動的に生成

## ■ 整ったフォークの配置を実現

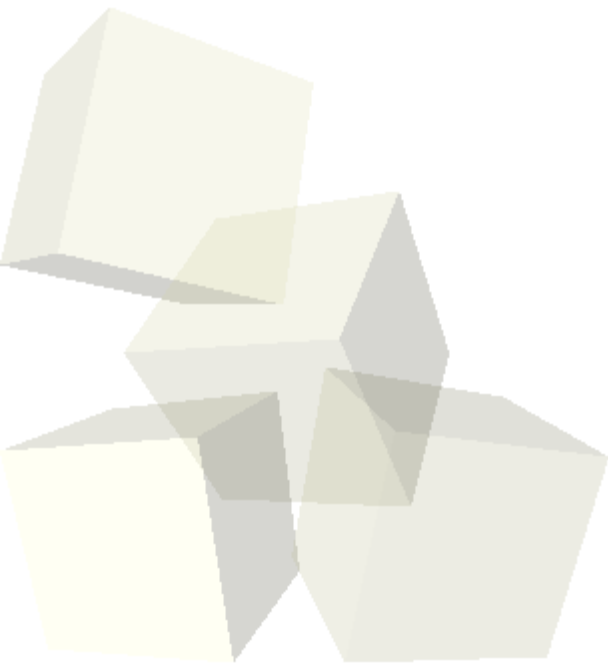
- ・ きれいに配置できる数学式を作成
- ・  $x1 = 20\cos(2\pi*i/n + \pi/2) + 300\cos(2\pi*i/n) + \text{circlet}$
- ...





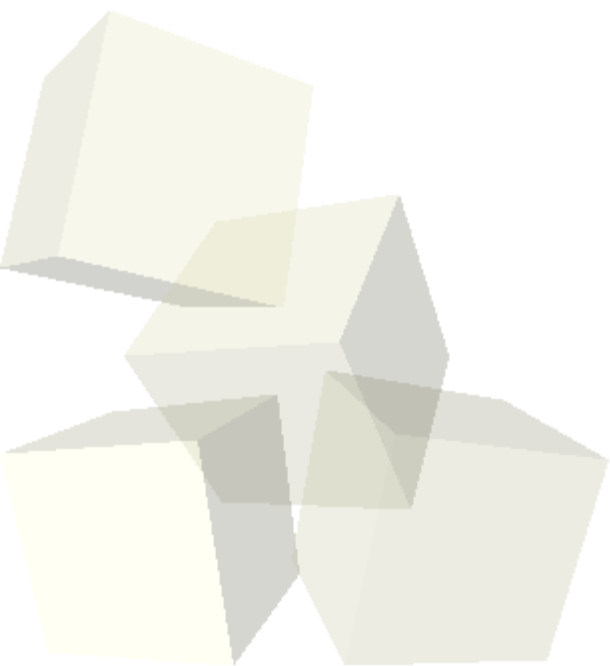
## ■ 状態を人ごとに用意することで競合を少なく

- ・ 競合解消戦略を考えるのではなく、そもそも競合が起こりづらく  
← 状態を細分化することで実装





## ■ 実際に動かしてみる



## ■ ちょっと見栄えが悪い

- ・ 記号(♂など)ではなく画像を使用することで改善  
→ フォークの向きなども表現可能に(?)

## ■ 哲学者の名前が固定

- ・ java.util.Setインタフェースを使用することで改善
- ・ 哲学者の配置を管理するメンバ変数を新たに用意



- 他のメンバーが記述したコードを読むのが大変
  - ・ クラスが複数存在すると、どこで何がされているかわかりにくい  
変数、メソッド名も適切なものが必要  
→ コメント大事!!!
- メンバーの進捗に合わせるのが大変
  - ・ 各自の予定等もあり、メンバーの進捗に差が出てしまう
  - ・ グループ全体の進捗が遅れてしまうことも
- C#大変
  - ・ ブロックの配置等、アイデアが湧きづらい



ご静聴ありがとうございました

