Assignment 2 - Distributed AI & Intelligent Agents

Perttu Jääskeläinen   Group 1   Gabriele Morello
perttuj@kth.se   morello@kth.se

November 23, 2022

Contents

1 Introduction

In this assignment, we extended our previous implementation of a festival simulation with the possibility to participate in auctions to buy items (since the festival is very popular, the organizers have decided to utilize the hype to earn more money).

In addition to patrolling the festival area and dancing to the music of their favorite artists, the guests will occasionally be notified of a new auction somewhere in the festival area through the organizers official mobile- app. They have a short time to respond if they're interested, after which they're expected to arrive at the auction.

The auctions are implemented using different strategies based on the actual merch stand - the ones implemented are Dutch- (start with an artificially high price, lowering it if nobody accepts the price), English (start with a low number, participants can make higher bids) and Sealed-bid auctions (guests can freely make a bet, and the one with the highest bid wins). All auctioneer types will run concurrently throughout the festival, occasionally announcing a new auction which the guests can participate in.

Each auctioneer sells different types of items - clothing, CD's, or VIP tickets. Each guest has some preference for what they would like to buy/participate in - and will participate in auctions accordingly.

2 Implementation

In order to ensure the festival functions as intended, some functionality was abbreviated from the code. We will only be going through the new code and species in this section - since it is assumed the reader is already familiar with the basic implementation from the previous assignment.

2.1 Basic implementation

The basic implementation includes the new species **Auctioneer** and **DutchAuctioneer** - where the latter is an extension of the former species. The **FestivalGuest** is also extended to be able to respond to and participate in auctions (with some new attributes and reflexes).

The **Auctioneer** species has some life-cycle reflexes to control the flow of auctions: **startNewAuction**, which periodically starts a new auction, the **notifyAuctionStart** to announce the new auction to festival guests, **collectParticipants** to register which guests are interested in the auction, **startAuction** to start the auction when guests arrive, and the **readProposals** to read/accept proposals from guests interested in the auction item for the announced price.

In addition to these reflexes, the **DutchAuctioneer** also has a **receiveRefuseMessages** reflex to handle cases where all participants refuse the price - which will cause the price to be lowered slightly, and a new round of bidding is done.

The **FestivalGuest** species is modified with the following reflexes: **answerInvitation** to either join new auctions or register that auctions have finished, **gotoAuction** to move toward the auction location, **respondToCfp** to respond to calls for proposal in auctions, and **receiveAccept/receiveReject—Proposals** to react to scenarios where proposals might be either accepted or rejected.

When not participating in auctions, guests will dance around the festival area and get food/drinks when necessary.

2.2 Challenge 1 - Interest in certain items

For the first challenge, we slightly modified the **FestivalGuest** and **Auctioneer** species to either sell/be interested in either **CD's**, **VIP Tickets** or **clothing**. Auctioneers would only sell items of one category, and guests would only participate in auctions that were selling items of that category.

This required small changes from the basic implementation - the initial **inform** FIPA call included which item type was for sale, and the **answerInvitation** reflex in the **FestivalGuest** was modified to only accept invitations to auctions that were of interest to each guest.

To better visualize these changes, the guests and auctioneers were given certain colors - **teal** for interest/selling **CD's**, **orange** for interest/selling of **clothing**, and **purple** for interest/selling of **VIP tickets**.

2.3 Challenge 2 - Different auction types

In addition to having interest in only certain auction types, the auctioneers would have different strategies for the items that were on sale. In addition to the **DutchAuctioneer**, we also implemented the **English-** and **SealedBidAuctioneer**:s, which have slightly different strategies.

The **EnglishAuctioneer** will start with a minimum price, which each guest can propose a new bid to. In each round, the current highest bid is announced, and rounds are repeated until only the highest bidder remains - who then wins the item for the bidded price.

For the **SealedBidAuctioneer**, each guests can submit *only one bid* - which should be as close to their personal value for the item. For this implementation, we assumed that every guest values the items they're after in an unlimited fashion - they will pay as much money as they have available. The person who submits the highest bid always wins the item in question.

We also did some comparisons of values for the different auctioneer types - in terms of how much money was spent, on average, for the items in question: **English**: ∼8959 (43 items), **Dutch**: ∼9973 (33 items), and **SealedBid**: ∼8200 (36 items) - we can clearly see that the **SealedBid** is the most beneficial for the guests, and both **Dutch** and **English** are beneficial for the organizers. Logs from a run using challenge 2 can be seen in figure 1.

3 Discussion

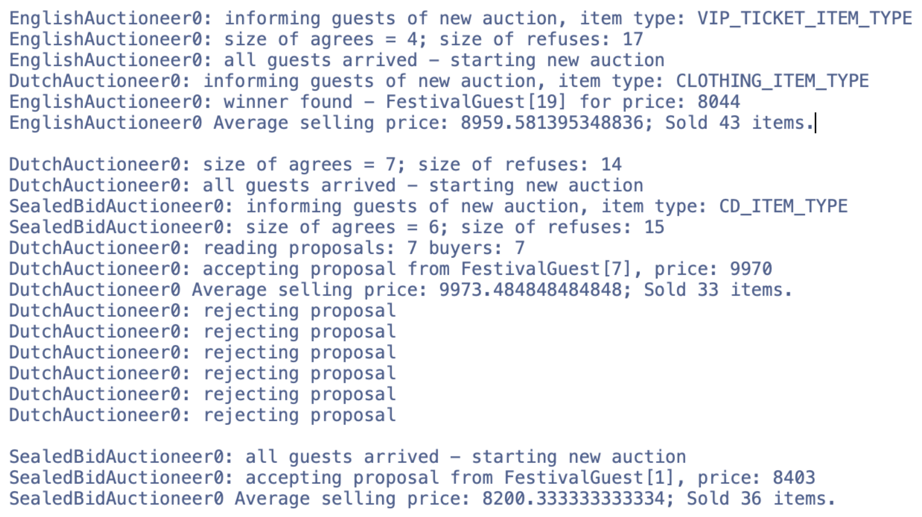
This homework introduced us to more complex communication in the GAMA-platform, which proved to be quite challenging. Having to handle many asynchronous flows was hard at first, but as with the first homework, once we understood the big picture and learned the API it became quite simple to follow.

Since we already intended to implement both challenges, we spent more time on the basic implementation than probably necessary if one was only doing the basic functionality - which resulted in the first challenge being quite easy to implement.

When implementing concurrent auctions, we had many problems. Most of them were related to the concurrent flows - since we're using the FIPA protocol, we ran into scenarios where we would receive concurrent auction announcement, which caused guests to abandon one auction and go for the next. This would result in strange behavior at the auctioneer, since it expects all participants to respond.

We also had some challenges with conditions for reflexes - we would join an auction and instantly leave it, since we had an old message from the previous auction ending queued, which was instantly run the moment we entered another auction.

In the end, it was beneficial for us since we gained lots of experience debugging the flow and simplifying our code, which worked well in the end.



```
EnglishAuctioneer0: informing guests of new auction, item type: VIP_TICKET_ITEM_TYPE
EnglishAuctioneer0: size of agrees = 4; size of refuses: 17
EnglishAuctioneer0: all guests arrived - starting new auction
DutchAuctioneer0: informing guests of new auction, item type: CLOTHING_ITEM_TYPE
EnglishAuctioneer0: winner found - FestivalGuest[19] for price: 8044
EnglishAuctioneer0 Average selling price: 8959.581395348836; Sold 43 items.

DutchAuctioneer0: size of agrees = 7; size of refuses: 14
DutchAuctioneer0: all guests arrived - starting new auction
SealedBidAuctioneer0: informing guests of new auction, item type: CD_ITEM_TYPE
SealedBidAuctioneer0: size of agrees = 6; size of refuses: 15
DutchAuctioneer0: reading proposals: 7 buyers: 7
DutchAuctioneer0: accepting proposal from FestivalGuest[7], price: 9970
DutchAuctioneer0 Average selling price: 9973.484848484848; Sold 33 items.
DutchAuctioneer0: rejecting proposal
DutchAuctioneer0: rejecting proposal
DutchAuctioneer0: rejecting proposal
DutchAuctioneer0: rejecting proposal
DutchAuctioneer0: rejecting proposal
DutchAuctioneer0: rejecting proposal

SealedBidAuctioneer0: all guests arrived - starting new auction
SealedBidAuctioneer0: accepting proposal from FestivalGuest[1], price: 8403
SealedBidAuctioneer0 Average selling price: 8200.333333333334; Sold 36 items.
```

Figure 1: *Some logs from the second challenge, where the auctioneers are printing out the average price for each item sold.*

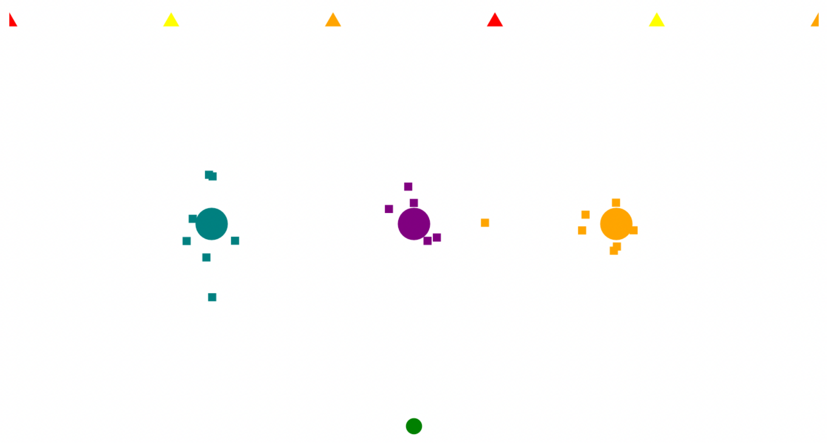


Figure 2: *Example run of a simulation, where all guests are currently participating in an auction.*