ID2209 – Distributed Artificial Intelligence and Intelligent Agents

Assignment 2 - Negotiation and Communication (FIPA)

Group 18

Paul Hübner
Samuel Horacek

24-11-2023

In this assignment, we were tasked with creating a simulation of an auction using the FIPA protocol. In the base assignment, this is a Dutch auction. In principle, an Auctioneer announces an auction. If there is interest from Bidders, the auction starts with the Auctioneer setting a price. The first Bidder willing to pay this price is the winner, and if no Bidder is willing to buy, the Auctioneer decreases the set price and another round of bidding proceeds. This repeats until a winner is found or the Auctioneer's minimal selling price is met.

We could set this simulation in the context of the previous "festival" assignment or start anew. We chose the latter, to start from a clean slate and no technical debt.

The bonuses look at having multiple auctions at the same time and implementation of different types of auctions - sealed and Vickrey in our implementation.

How to Run

Run GAMA (version 1.9 recommended) and import the Exercise2 folder (including *Exercise2.gaml*, *Exercise2_Bonus1.gaml* and *Exercis21_Bonus2.gaml* models) as a new project. Press main to run the simulation.

The parameters in *global* can be tweaked to change the number of bidders or the number of concurrent auctions.

Species

Auctioneer

Auctioneer is the initiator of auctions. It is responsible for announcing new auctions, and new rounds of bidding. Subsequently, it either accepts or rejects every proposal/bid.
In the base assignment, the Auctioneer is the only one (and thus there is only one auction at a time) and implements a Dutch auction.

In the second bonus, we generalised the Auctioneer species and introduced subspecies for each approach to auctioning - DutchAuctioneer, SealedAuctioneer, and VickreyAuctioneer. The generic Auctioneer holds the common auctioning logic, whereas the rest implement a specific type of auction.

Bidder

The Bidder plays a predominantly passive role in this simulation. It is waiting to receive invitations to auctions and to bid when possible. If its bid is rejected, nothing happens, but when its bid is accepted, it sends a confirmation *Inform* message to confirm it is the winner of the auction (and thereby provide its shipping address).

Implementation

We first implemented a skeleton of Auctioneer and Bidder on top of the festival guest logic. However, this became a bit bloated and hard to work with, so we factored it out into its own thing.

We developed the Auctioneer and then the Bidder. We focused on the implementation of the main event loop of the Auctioneer as a basis. As we built this up, we added functionality to the Bidder.

Then, we performed another set of refactoring where we introduced the ability to decide if you want to join/leave an auction, and refined some of the code. This was necessary to then make the bonuses.

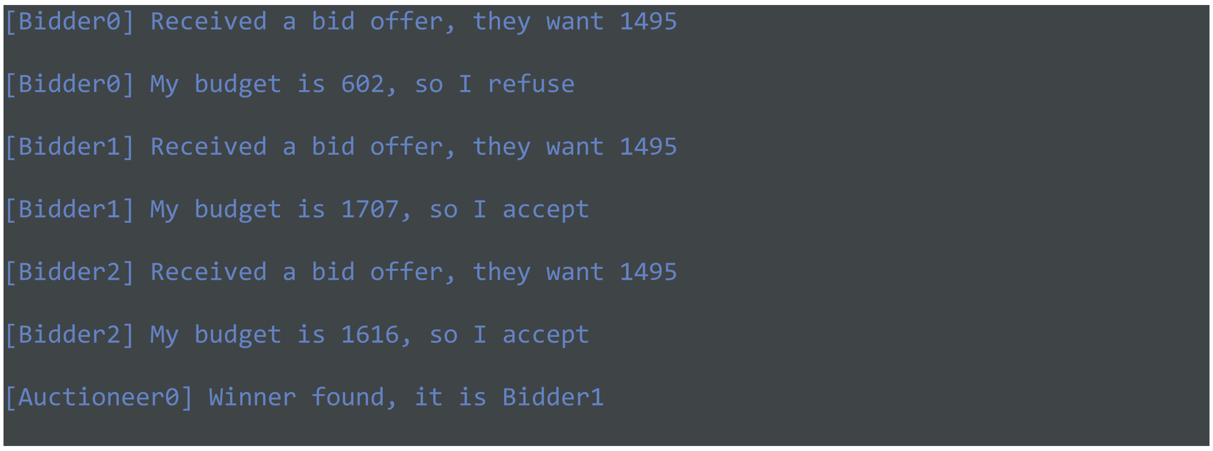
One thing to note is that the bidder has a budget representing how much they want to pay for an item. This budget never decreases for simplicity.

Results

Although we have some visual representation, the results are best inspected using the logs of the ongoing auction. In this example of the base assignment, we have 3 Bidders and a single Auctioneer (Dutch auction).

As is visible from the following logs, the auction starts at the price of 1745. The winner is not found in the first round of bidding, but only in the second (as the initial price of 1745 is decreased to 1495, which 2 of the Bidders can afford). In the end, it is Bidder1 who is pronounced winner, as he was first to pay the desired price.

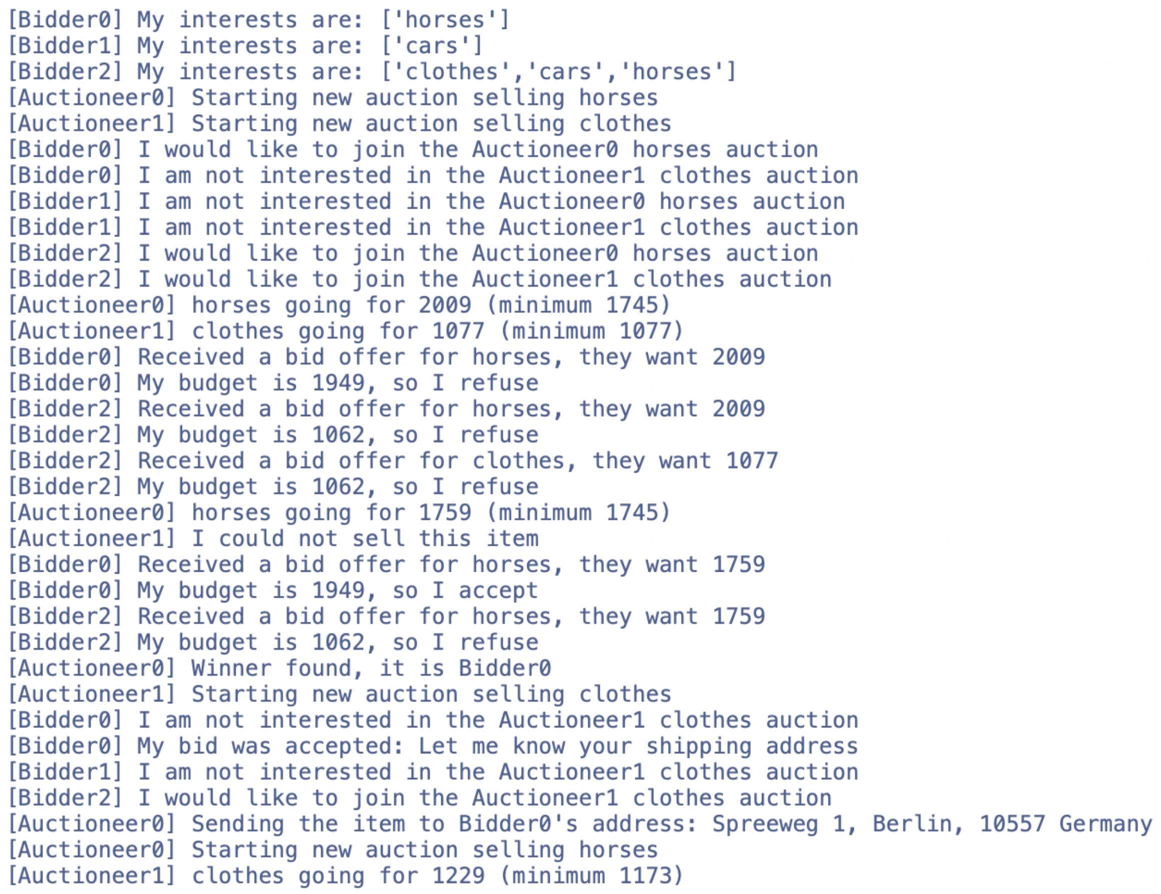
```
[Bidder0] Received word of a new auction, I would like to join!

[Bidder1] Received word of a new auction, I would like to join!

[Bidder2] Received word of a new auction, I would like to join!

[Auctioneer0] Going for 1745 (minimum 1172)

[Bidder0] Received a bid offer, they want 1745

[Bidder0] My budget is 602, so I refuse

[Bidder1] Received a bid offer, they want 1745

[Bidder1] My budget is 1707, so I refuse

[Bidder2] Received a bid offer, they want 1745

[Bidder2] My budget is 1616, so I refuse

[Auctioneer0] Going for 1495 (minimum 1172)
```



```
[Bidder0] Received a bid offer, they want 1495

[Bidder0] My budget is 602, so I refuse

[Bidder1] Received a bid offer, they want 1495

[Bidder1] My budget is 1707, so I accept

[Bidder2] Received a bid offer, they want 1495

[Bidder2] My budget is 1616, so I accept

[Auctioneer0] Winner found, it is Bidder1
```

Challenge 1

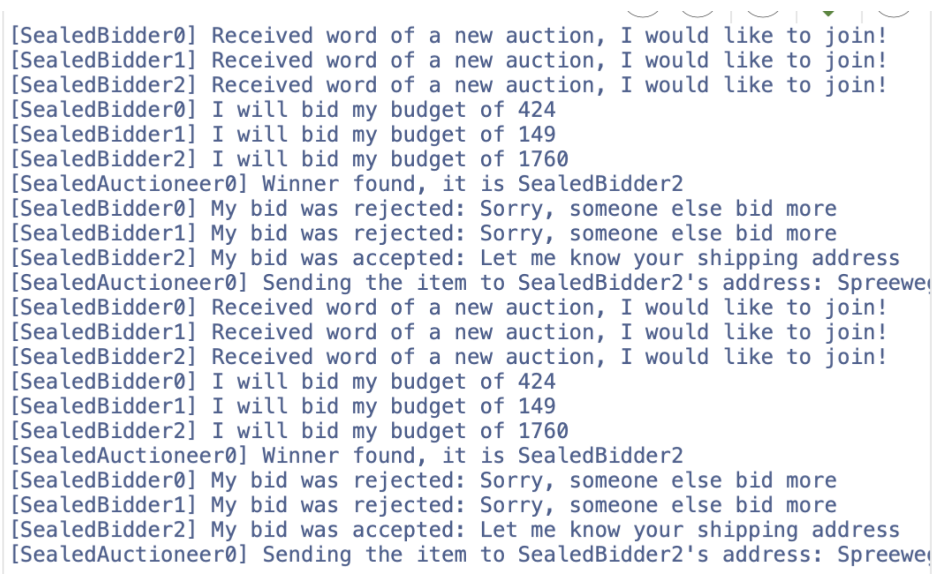
The first challenge was quite straightforward. We just had to spawn in multiple Auctioneers, and join/reject the auctions depending on if the Bidders were interested in the theme. Below is a screenshot of two concurrent auctions going on, note that when one completes the next will start, this may even happen in the middle of another auction.



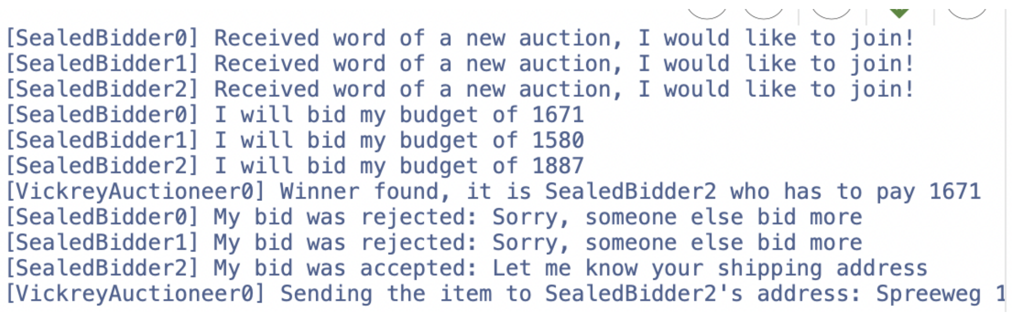
```
[Bidder0] My interests are: ['horses']
[Bidder1] My interests are: ['cars']
[Bidder2] My interests are: ['clothes','cars','horses']
[Auctioneer0] Starting new auction selling horses
[Auctioneer1] Starting new auction selling clothes
[Bidder0] I would like to join the Auctioneer0 horses auction
[Bidder0] I am not interested in the Auctioneer1 clothes auction
[Bidder1] I am not interested in the Auctioneer0 horses auction
[Bidder1] I am not interested in the Auctioneer1 clothes auction
[Bidder2] I would like to join the Auctioneer0 horses auction
[Bidder2] I would like to join the Auctioneer1 clothes auction
[Auctioneer0] horses going for 2009 (minimum 1745)
[Auctioneer1] clothes going for 1077 (minimum 1077)
[Bidder0] Received a bid offer for horses, they want 2009
[Bidder0] My budget is 1949, so I refuse
[Bidder2] Received a bid offer for horses, they want 2009
[Bidder2] My budget is 1062, so I refuse
[Bidder2] Received a bid offer for clothes, they want 1077
[Bidder2] My budget is 1062, so I refuse
[Auctioneer0] horses going for 1759 (minimum 1745)
[Auctioneer1] I could not sell this item
[Bidder0] Received a bid offer for horses, they want 1759
[Bidder0] My budget is 1949, so I accept
[Bidder2] Received a bid offer for horses, they want 1759
[Bidder2] My budget is 1062, so I refuse
[Auctioneer0] Winner found, it is Bidder0
[Auctioneer1] Starting new auction selling clothes
[Bidder0] I am not interested in the Auctioneer1 clothes auction
[Bidder0] My bid was accepted: Let me know your shipping address
[Bidder1] I am not interested in the Auctioneer1 clothes auction
[Bidder2] I would like to join the Auctioneer1 clothes auction
[Auctioneer0] Sending the item to Bidder0's address: Spreeweg 1, Berlin, 10557 Germany
[Auctioneer0] Starting new auction selling horses
[Auctioneer1] clothes going for 1229 (minimum 1173)
```

Challenge 2

The second challenge was more hands-on, as we had to factor out common logic and abstract it to be able to create an Auctioneer hierarchy. This resulted in quite a lot of classes, but thankfully due to our many refactors we did not have to change any of the core control flow! Below are screenshots of a Sealed and Vickrey auction.



```
[SealedBidder0] Received word of a new auction, I would like to join!
[SealedBidder1] Received word of a new auction, I would like to join!
[SealedBidder2] Received word of a new auction, I would like to join!
[SealedBidder0] I will bid my budget of 424
[SealedBidder1] I will bid my budget of 149
[SealedBidder2] I will bid my budget of 1760
[SealedAuctioneer0] Winner found, it is SealedBidder2
[SealedBidder0] My bid was rejected: Sorry, someone else bid more
[SealedBidder1] My bid was rejected: Sorry, someone else bid more
[SealedBidder2] My bid was accepted: Let me know your shipping address
[SealedAuctioneer0] Sending the item to SealedBidder2's address: Spreewe
[SealedBidder0] Received word of a new auction, I would like to join!
[SealedBidder1] Received word of a new auction, I would like to join!
[SealedBidder2] Received word of a new auction, I would like to join!
[SealedBidder0] I will bid my budget of 424
[SealedBidder1] I will bid my budget of 149
[SealedBidder2] I will bid my budget of 1760
[SealedAuctioneer0] Winner found, it is SealedBidder2
[SealedBidder0] My bid was rejected: Sorry, someone else bid more
[SealedBidder1] My bid was rejected: Sorry, someone else bid more
[SealedBidder2] My bid was accepted: Let me know your shipping address
[SealedAuctioneer0] Sending the item to SealedBidder2's address: Spreewe
```



```
[SealedBidder0] Received word of a new auction, I would like to join!
[SealedBidder1] Received word of a new auction, I would like to join!
[SealedBidder2] Received word of a new auction, I would like to join!
[SealedBidder0] I will bid my budget of 1671
[SealedBidder1] I will bid my budget of 1580
[SealedBidder2] I will bid my budget of 1887
[VickreyAuctioneer0] Winner found, it is SealedBidder2 who has to pay 1671
[SealedBidder0] My bid was rejected: Sorry, someone else bid more
[SealedBidder1] My bid was rejected: Sorry, someone else bid more
[SealedBidder2] My bid was accepted: Let me know your shipping address
[VickreyAuctioneer0] Sending the item to SealedBidder2's address: Spreeweg 1
```

Creative Implementation

We did not implement creative features for this assignment.

Discussion/Conclusion

The base assignment itself was not that hard to implement. We went through the documentation and conceptual overview of the FIPA protocol. The actual implementation was harder, especially the refactoring needed to implement the 2 bonuses.

Overall, the assignment was insightful as it showcased to us how FIPA can be used and especially how the reflex event loop works, as it was the part of the assignment we had the most issues with. Such as certain updates/events occurring before ones we expected.