Life cycle of a HEP:

1. Homeowner triggers its creation
2. New HEP, worth $10,000, is auctioned to lowest (in equity percentage) bidder. Equity percentage is now fixed for the life of the HEP.
3. Investor decides to sell, makes the HEP available on the secondary market
4. Secondary market auction, HEP sold to highest bidder.
5. … Repeat 3, 4 …
6. Homeowner sells the underlying real property and the claim “matures”
7. Investor is paid final claim value Cf = S­f - M0 (Sf : final sale price of the home; M0 : initial mortgage amount). We use M0 because the homeowner is reimbursed all mortgage principal payments from the sale proceeds before the remainder is divided.

There are a few technical challenges here, especially in the GoldSim implementation. The obvious way to model a HEP is with an object-oriented programming paradigm, which GoldSim does not support.

The current solution involves lots and lots of array elements. Each abstract HEP object is given an ID number, as is each investor, each home, and each neighborhood. Then, rather than creating a GoldSim element with attributes “underlying home”, “current owner”, etc, (which is pretty much impossible – but see below), we create arrays called “Home”, “Owner”, etc. Then investor *k* owns HEP *j*, and so forth.

This is cumbersome, but it works – mostly. GoldSim arrays are fixed in size, so it is necessary to allocate enough space for the maximum number of HEPs, investors, etc, that you might want in your simulation. Array sizes are fixed in many low-level programming languages, and this makes for excellent efficiency in execution, but GoldSim is not a low-level programming language. The software attempts to keep every element continually up-to-date, even during the programming process, and does not know which elements represent “live” HEPS and which are empty space. My computer crashed 3 times on Thursday before I figured this out and cranked the array sizes way down. It won’t hurt our early simulations, but it does but a limit on the biggest simulation my poor laptop can run.

There is another way we could implement things; I thought of it after I had put a few hours into the array-based implementation and didn’t want to backtrack, but it’s worth considering for later. There are two ways to sneak a dynamically sized array into Goldsim: discrete change queues and dynamically linked spreadsheets. Each would present a few challenges of its own, but a dynamically sized list would allow us to handle each HEP as a discrete object, and would curtail the processing waste of the current model.

For now, we have the static-arrays implementation, and the “first draft” of the primary auction model is complete (but untested and surely buggy). The secondary auction model is easy: we can reuse the primary model, tweaking the arithmetic a bit to reflect the change in bid structure.

Details:

1. HEP data structures
   1. Owner: the investor who currently owns it.
   2. Home
   3. Percentage; the equity claim percentage.
   4. Price History; tracks last secondary market sale price
   5. needsPrimary; triggers a primary auction of the hep
   6. needsSecondary; triggers a secondary auction
   7. Valuations; a HEPs x Investors matrix containing the amount each investor would pay for each HEP.
2. Investor data structures
   1. ?
   2. ?
3. Auction Model
   1. An auction queue script monitors the needsPrimary array, puts the ID of a HEP that needs an auction into the first empty auction ID (auctions are also modeled with a bunch of arrays).
   2. Based on fixed HEP primary price of $10,000, calculate each investor’s bid in percentage points, rounded to highest 0.1%. If there is a single winning bidder, then the winning bid is equal to the *second* highest bidder’s best bid, minus 0.1 percentage points.
   3. Record lowest bids in today’s bidding.
   4. In case of a tie in valuations, the winning bidder is whomever happened to get the bid in first: choose randomly.
   5. Record today’s winning bidders
   6. If today’s lowest bid in a particular auction is unchanged from yesterday’s lowest bid, then the auction closes. Update the Owner list, the Percentage list, the list of Live Auctions.
   7. Otherwise, bidding will continue tomorrow – based on investor valuations recalculated with information available tomorrow.