ALL_VARS.pro

Define variables relevant for ultrasound. I think right now `Npulse` and `PulseGap` are irrelevant.

```
///////// Ultrasound Vars /////////
declare int StimInterval;
declare int StimCond;
declare int LastStim;
declare int Npulse;
declare int PulseGap;
```

DEFAULT.pro

Code block for each individual animal in the *search task specific* section. Assign values to the US variables. `Npulse` and `PulseGap` should be irrelevant.

```
/// Ultrasound vars /////
VarEcc       = 0;      // 0 = off, 1 = on;
                       // variable eccentricity from list line 137 LOC_RAND.pro
LatStruct    = 1;      // For US detection task:
                       // 0 = search items only at 4 corners;
                       // 1 = normal search, all locations
Npulse       = 600;    // number of pulses sent
PulseGap     = 1000;   // gap between pulses
StimInterval = 600000; // 10 minutes = 600000ms
StimCond     = 1;      // 0 = stim starting block 1 (min 0),
                       // 1 = stim starting block 2 (min 10)
```

EVENTDEF.pro

Define encodes for the stimulation. `Stimulation_` and `EndStim_` are needed.

```
declare hide constant ShamStim_         = 665;
declare hide constant Stimulation_      = 666;
declare hide constant EndStim_          = 667;
```

LOC_RAND.pro

In the function `process LOC_RAND` is a conditional block to determine target location. For ultrasound, we used an option where the positions at the vertical meridian were not used as target position (as defined by the `LatStruct` variable).

```
if (SingMode == 0)
{
  d1color = 250;
  if (ArrStruct == 1) // structured array mode
    {
    if (TargTrainSet == 1) // random target location on each trial
      {
          if (LatStruct == 0) // here LatStruct == 0 presents items ONLY at the 4 square location
s; developed for ultrasound
```

```
        {
        TgAng = Random(6);

        if (TgAng == 0)
          {
          THemi = 1;
          Rand_targ_angle = Angle_list[1];
          Rand_d1_angle = Angle_list[4];    //See DEFAULT.pros for setting each of these variable
s
          Rand_d2_angle = Angle_list[2];
          Rand_d3_angle = Angle_list[6];
          Rand_d4_angle = Angle_list[0];    //See DEFAULT.pros for setting each of these variable
s
          Rand_d5_angle = Angle_list[3];
          Rand_d6_angle = Angle_list[5];
          Rand_d7_angle = Angle_list[7];
          }
        else if (TgAng == 1)
          {
          THemi = 1;
          Rand_targ_angle = Angle_list[2];
          Rand_d1_angle = Angle_list[5];    //See DEFAULT.pros for setting each of these variable
s
          Rand_d2_angle = Angle_list[1];
          Rand_d3_angle = Angle_list[7];
          Rand_d4_angle = Angle_list[3];    //See DEFAULT.pros for setting each of these variable
s
          Rand_d5_angle = Angle_list[4];
          Rand_d6_angle = Angle_list[6];
          Rand_d7_angle = Angle_list[0];
          }
        else if (TgAng == 2)
          {
          THemi = 1;
          Rand_targ_angle = Angle_list[3];
          Rand_d1_angle = Angle_list[6];    //See DEFAULT.pros for setting each of these variable
s
          Rand_d2_angle = Angle_list[4];
          Rand_d3_angle = Angle_list[0];
          Rand_d4_angle = Angle_list[1];    //See DEFAULT.pros for setting each of these variable
s
          Rand_d5_angle = Angle_list[5];
          Rand_d6_angle = Angle_list[2];
          Rand_d7_angle = Angle_list[7];
          }
        else if (TgAng == 3)
          {
          THemi = 2;
          Rand_targ_angle = Angle_list[5];
          Rand_d1_angle = Angle_list[3];    //See DEFAULT.pros for setting each of these variable
s
          Rand_d2_angle = Angle_list[7];
          Rand_d3_angle = Angle_list[1];
          Rand_d4_angle = Angle_list[2];    //See DEFAULT.pros for setting each of these variable
s
          Rand_d5_angle = Angle_list[4];
          Rand_d6_angle = Angle_list[6];
          Rand_d7_angle = Angle_list[0];
          }
```

```
        else if (TgAng == 4)
          {
          THemi = 2;
          Rand_targ_angle = Angle_list[6];
          Rand_d1_angle = Angle_list[5];    //See DEFAULT.pros for setting each of these variable
s
          Rand_d2_angle = Angle_list[4];
          Rand_d3_angle = Angle_list[0];
          Rand_d4_angle = Angle_list[1];    //See DEFAULT.pros for setting each of these variable
s
          Rand_d5_angle = Angle_list[3];
          Rand_d6_angle = Angle_list[2];
          Rand_d7_angle = Angle_list[7];
          }
        else if (TgAng == 5)
          {
          THemi = 2;
          Rand_targ_angle = Angle_list[7];
          Rand_d1_angle = Angle_list[3];    //See DEFAULT.pros for setting each of these variable
s
          Rand_d2_angle = Angle_list[5];
          Rand_d3_angle = Angle_list[1];
          Rand_d4_angle = Angle_list[2];    //See DEFAULT.pros for setting each of these variable
s
          Rand_d5_angle = Angle_list[4];
          Rand_d6_angle = Angle_list[6];
          Rand_d7_angle = Angle_list[0];
          }
          }
          if (LatStruct == 1)
            {
              TgAng = Random(8);

            if (TgAng == 0)
              {
              THemi = 1;
              Rand_targ_angle = Angle_list[1];
              Rand_d1_angle = Angle_list[4];    //See DEFAULT.pros for setting each of these vari
ables
              Rand_d2_angle = Angle_list[2];
              Rand_d3_angle = Angle_list[6];
              Rand_d4_angle = Angle_list[0];    //See DEFAULT.pros for setting each of these vari
ables
              Rand_d5_angle = Angle_list[3];
              Rand_d6_angle = Angle_list[5];
              Rand_d7_angle = Angle_list[7];
              }
            else if (TgAng == 1)
              {
              THemi = 1;
              Rand_targ_angle = Angle_list[3];
              Rand_d1_angle = Angle_list[5];    //See DEFAULT.pros for setting each of these vari
ables
              Rand_d2_angle = Angle_list[1];
              Rand_d3_angle = Angle_list[7];
              Rand_d4_angle = Angle_list[2];    //See DEFAULT.pros for setting each of these vari
ables
              Rand_d5_angle = Angle_list[4];
              Rand_d6_angle = Angle_list[6];
              Rand_d7_angle = Angle_list[0];
```

```
              }
            else if (TgAng == 2)
              {
              THemi = 1;
              Rand_targ_angle = Angle_list[5];
              Rand_d1_angle = Angle_list[6];    //See DEFAULT.pros for setting each of these vari
ables
              Rand_d2_angle = Angle_list[4];
              Rand_d3_angle = Angle_list[0];
              Rand_d4_angle = Angle_list[1];    //See DEFAULT.pros for setting each of these vari
ables
              Rand_d5_angle = Angle_list[3];
              Rand_d6_angle = Angle_list[2];
              Rand_d7_angle = Angle_list[7];
              }
            else if (TgAng == 3)
              {
              THemi = 2;
              Rand_targ_angle = Angle_list[7];
              Rand_d1_angle = Angle_list[3];    //See DEFAULT.pros for setting each of these vari
ables
              Rand_d2_angle = Angle_list[5];
              Rand_d3_angle = Angle_list[1];
              Rand_d4_angle = Angle_list[2];    //See DEFAULT.pros for setting each of these vari
ables
              Rand_d5_angle = Angle_list[4];
              Rand_d6_angle = Angle_list[6];
              Rand_d7_angle = Angle_list[0];
              }
            else if (TgAng == 4)
              {
              THemi = 0;
              Rand_targ_angle = Angle_list[0];
              Rand_d1_angle = Angle_list[4];    //See DEFAULT.pros for setting each of these vari
ables
              Rand_d2_angle = Angle_list[2];
              Rand_d3_angle = Angle_list[6];
              Rand_d4_angle = Angle_list[1];    //See DEFAULT.pros for setting each of these vari
ables
              Rand_d5_angle = Angle_list[3];
              Rand_d6_angle = Angle_list[5];
              Rand_d7_angle = Angle_list[7];
              }
            else if (TgAng == 5)
              {
              THemi = 1;
              Rand_targ_angle = Angle_list[2];
              Rand_d1_angle = Angle_list[1];    //See DEFAULT.pros for setting each of these vari
ables
              Rand_d2_angle = Angle_list[7];
              Rand_d3_angle = Angle_list[3];
              Rand_d4_angle = Angle_list[5];    //See DEFAULT.pros for setting each of these vari
ables
              Rand_d5_angle = Angle_list[4];
              Rand_d6_angle = Angle_list[6];
              Rand_d7_angle = Angle_list[0];
              }
            else if (TgAng == 6)
              {
              THemi = 0;
```

```
              Rand_targ_angle = Angle_list[4];
              Rand_d1_angle = Angle_list[2];    //See DEFAULT.pros for setting each of these vari
ables
              Rand_d2_angle = Angle_list[0];
              Rand_d3_angle = Angle_list[6];
              Rand_d4_angle = Angle_list[1];    //See DEFAULT.pros for setting each of these vari
ables
              Rand_d5_angle = Angle_list[3];
              Rand_d6_angle = Angle_list[5];
              Rand_d7_angle = Angle_list[7];
              }
            else if (TgAng == 7)
              {
              THemi = 2;
              Rand_targ_angle = Angle_list[6];
              Rand_d1_angle = Angle_list[3];    //See DEFAULT.pros for setting each of these vari
ables
              Rand_d2_angle = Angle_list[1];
              Rand_d3_angle = Angle_list[5];
              Rand_d4_angle = Angle_list[2];    //See DEFAULT.pros for setting each of these vari
ables
              Rand_d5_angle = Angle_list[4];
              Rand_d6_angle = Angle_list[0];
              Rand_d7_angle = Angle_list[7];
              }
            }
          }
```

SETS_TRL.pro

Declare a variable that controls US stimulation:

```
declare hide int StimTm;    // Should we stim on this trial?
```

Select the stimulation period. Since it is not happening at variable task periods, only one value is used.

```
// 7) Choose whether to stim
//StimTm = Random(2); //allows us to randomize the time stim is delivered; see task stages in SCH
TRIAL.pro
StimTm = 1; //Single stim time
//StimTm = 0; //stim off
//StimTm = 5; //For prolonged stim protocol
```

INFOS.pro

We encoded the timing of the US stimulation. However, for now we use a constant time and therefore this entry will be constant and not really ne needed to encoded as strobe. But we will need a strobe that tells if the current trial was an US or non-US trial (even thugh we have blocks of US stimulation).

In `if (State == run_search_sess)`:

```
Event_fifo[Set_event] = StimTm + 5100;                          // Send event and...
```

```
Set_event = (Set_event + 1) % Event_fifo_N;                    // ...incriment event queu
```

SCHTRIAL.pro

Declaration of `StimDone` variable.

```
// Stim complete?
declare hide int    StimDone;
StimDone = 0;
```

When animal is fixating we checked if it is about time to trigger US stimulation. We implemented several options as defined by the variable `StimTm`. However, we ended up using a single stimulation that was triggered prior to target onset. Thus, for now we should be able to reduce the code for exactly this condition. For this, we could ommitt the `StimTm` variable.

in `else if (stage == fixating_ph)`:

```
else if (StimDone == 0 && StimTm == 1 && In_FixWin && time() > aquire_fix_time + (curr_holdtime -
 150)) // But if the eyes are still in the window at end of holdtime...
  {
    printf("Going to Stimulate on Line 235...\n");
    spawn STIM(stim_channel);
    StimDone = 1;
  }
```

I do not list here the other code blocks for different `StimTm` values because this is not needed at the moment.

SEARCH.pro

Here, we controlled controlled the timing of US blocks

Declare a variable that sets the last stim to zero in `process SEARCH()`:

```
LastStim               = 0;
```

in 'while (state == run_search_sess)':

```
if (time() > (LastStim + StimInterval)) // for blocked, on-task stimulation
 {
 LastStim = time();

 if(StimCond == 1)
   {
   Event_fifo[Set_event] = ShamStim_;
   Set_event = (Set_event + 1) % Event_fifo_N;

   StimCond = 0;
   //StimTm = 0;// for trial-based, blocked stimulation vs. non-stimulation

   //wait(Npulse * PulseGap); // Provides TTL stim parameters for blocked, pre-task stimulation;
```

```
here, this is sham and simply waits

   Event_fifo[Set_event] = EndStim_;
   Set_event = (Set_event + 1) % Event_fifo_N;
   }
 else
   {
   Event_fifo[Set_event] = Stimulation_;
   Set_event = (Set_event + 1) % Event_fifo_N;

   StimCond = 1;
   pcnt = 0;
   printf("Stim Time%d",StimTm);
/*              while (pcnt < Npulse) // Provides TTL stim parameters for blocked, pre-task stimu
lation; here, this stims continuously based on DEFAULT.pro settings
     {
     spawn STIM(stim_channel);
     wait(PulseGap);
     pcnt = pcnt+1;
     } */

   //StimTm = 1;    // for trial-based, blocked stimulation vs. non-stimulation

   Event_fifo[Set_event] = EndStim_;
   Set_event = (Set_event + 1) % Event_fifo_N;
   }
 }
```

The second conditional block with the pulsed stimulation might not be needed.