John Romero Programming Proverbs

- 10. “Try to code transparently. Tell your lead and peers exactly how you are going to solve your current task and get feedback and advice. Do not treat game programming like each coder is a black box. The project could go off the rails and cause delays.”

- John Romero, “The Early Days of Id Software - John Romero @ WeAreDevelopers Conference 2017”

Implementing labels in Doom3

- so far we have implemented labels in chisel
  - and the bot can look up the label position

- it would be useful to allow the bot to test if a label is in it field of view (FOV)
  - we get the doom3 engine to determine the FOV
  - fortunately labels (and items, players) are contained as entities in the map file
  - the doom3 engine can be queried about entities

Example: section of a .map file

$HOME/Sandpit/chisel/python/tiny.map

```
// entity 1
{
    "classname" "info_player_start"
    "name" "info_player_start_1"
    "origin" "-120.000000 -72.000000 12.000000"
    "angle" "180"
}
// entity 2
{
    "classname" "python_doommarine_mp"
    "name" "python_doommarine_mp_1"
    "anim" "idle"
    "origin" "-552.000000 -72.000000 12.000000"
    "ambush" "1"
}
...
// entity 40
{
    "classname" "item_default"
    "name" "label_1"
    "label" "the_ammo_loc"
    "origin" "-1320.000000 -312.000000 12.000000"
}
```

Points of interest about the .map file

- do not edit it - unless you know what you are doing!
- each asset entity has a `name` and an `origin`
  - an asset is anything other than a wall, ceiling, floor
  - for example human player, python bot, monster, ammo, label, weapon, armour, health pack, light
- entity 0 contains all walls, ceiling, floors, beams etc
- entity 1 is always the human player
  - pybots are numbered after the human player ie entities 2 upwards

Points of interest about the .map file

- thereafter each asset has its own entity
  - hint examine `$HOME/Sandpit/chisel/python/tiny.map`
- for the purpose of this coursework ensure that your spawn point is not on the same axis as the Python bot
- make sure you have successfully completed last weeks tutorial (adding labels to chisel)

$HOME/Sandpit/chisel/python/pen2map.py

- make sure your label entity has a name

$HOME/Sandpit/chisel/python/pen2map.py

$HOME/Sandpit/chisel/python/pen2map.py

```
def generateLabels (o, e):
    n = 1
    for r in list(rooms.keys ()):
        if debugging:
            print (rooms[r].labels)
        for label_desc, xy in rooms[r].labels:
            o.write ("// entity " + str (e) + '\n')
            o.write ("{\n")
            o.write ('    "classname" "item_default"\n')
            o.write ('    "name" "label_' + str (n) + '"\n')
            o.write ('    "label" "' + label_desc + '"\n')
            o.write ('    "origin" "')
            xyz = toIntList (xy) + [-invSpawnHeight]
            xyz = subVec (xyz, [minx, miny, getFloorLevel (r)])
            v = midReposition (xyz)
            o.write ('%f %f %f"\n' % (v[0], v[1], v[2]))
            o.write ("}\n")
            n += 1
            e += 1
    return o, e
```

Entity gotya

- the entities in the `.map` file are different to the runtime entities in the doom3 engine
- many times there is a one to one correspondence (human player is always entity 1)
- the runtime `entities` array also contains player weapon model entities and torso component entities and others
- this means that a label entity read from the map file might have a different runtime entity number

Entity gotya

- therefore we need a way to translate between map entities and runtime entities

- the example code `$HOME/Sandpit/git-doom/pybot/python_doommarine_find_label.py` is the same as `$HOME/Sandpit/git-doom/pybot/python_doommarine.py` except the function `botMain` has the following code:

Example bot test code

- **$HOME/Sandpit/git-doom/pybot/python_doommarine_find_label.py**

```
def botMain (b):
    global me
    print("success!  python doom marine is alive")

    print("trying to get my id...", end=' ')
    me = b.me ()
    print("yes")
    print("the python marine id is", me)

    #    an ugly hack to test entity label lookup
    label_entity_map = b._cache.getEntityNo ("label", "the_ammo_loc")
    print ("map entity number of label =", label_entity_map)
    print ("map entity position =", b._cache.getEntityPos (label_entity_map))
```

Example bot test code

$HOME/Sandpit/git-doom/pybot/python_doommarine_find_label.py

```
    #    end of an ugly hack to test entity label lookup
    while True:
        print ("map label", label_entity_map)
        label_entity_no = b._cache._basic.mapToRunTimeEntity (label_entity_map)
        print ("runtime entity label", label_entity_no)
        b._cache.reset ()
        print ("can I see", label_entity_no, "?")
        if b.isvisible (label_entity_no):
            print ("yes I can see the label")
            b.face (label_entity_no)
            print ("moving towards the label")
            moveTowards (label_entity_no)
            time.sleep (5)
            moveTowards (1)
            time.sleep (5)
        else:
            print ("no I cannot")
            b.face (1)   # turn towards human player
        time.sleep (1)
```

Example bot test code

- notice the use of `label_entity_map` and `label_entity_no` which is derived by using the function `mapToRunTimeEntity`
- it should be possible to clean up this API, but for now it allows for experimentation

Example bot test code

- **$HOME/Sandpit/git-doom3/pybot-dhewm3/neo/game/Game_local.h**

```
    int MapToRunTimeEntity (int id);  // (convert map to runtime entity)
```

Example bot test code

- **$HOME/Sandpit/git-doom3/pybot-dhewm3/neo/game/Game_local.cpp**

```
int idGameLocal::MapToRunTimeEntity (int id)
{
  idMapEntity *mapEnt = mapFile->GetEntity (id);
  if (mapEnt && mapEnt->epairs.FindKey ("name"))
    {
      idStr name = mapEnt->epairs.FindKey ("name")->GetKey ();
      for (int i = 0; i < num_entities; i++)
    if (entities[i] && (idStr::Cmp (name, entities[i]->name) == 0))
      return i;
    }
  return -1;
}
```

Example bot test code

$HOME/Sandpit/git-doom3/pybot-dhewm3/neo/game/ai/pybot.cpp

```
void pyBotClass::interpretRemoteProcedureCall (char *data)
...
   else if (idStr::Cmpn (data, "change_weapon ", 14) == 0)
     rpcChangeWeapon (&data[14]);
   else if (idStr::Cmpn (data, "map_to_runtime_entity ", 21) == 0)
     rpcMapToRunTimeEntity (&data[21]);
...
```

$HOME/Sandpit/git-doom3/pybot-dhewm3/python-bot/botbasic.py

```
    #
    #  mapToRunTimeEntity - converts a static map entity into a dynamic entity.
    #

    def mapToRunTimeEntity (self, entity_no):
        if debug_protocol:
            print("requesting map_to_runtime_entity", entity_no)
        l = "map_to_runtime_entity %d \n" % (entity_no)
        self.s.send (l.encode ('utf-8'))
        l = self.getLine ()
        if debug_protocol:
            print("doom returned", l)
        return int (l)
```

- at this point the Python bot should be able to lookup a label and convert it into a runtime entity

- it would be good to give the bot the ability to test whether it can see a runtime entity

isvisible

$HOME/Sandpit/git-doom3/pybot-dhewm3/python-bot/botbasic.py

```
    #
    #  isvisible - returns True if entity_no is visible by the bot.
    #

    def isvisible (self, entity_no):
        if entity_no >= 0:
            if debug_protocol:
                print("requesting canSeeEntity", entity_no)
            l = "can_see_entity %d \n" % (entity_no)
            self.s.send (l.encode ('utf-8'))
            l = self.getLine ()
            if debug_protocol:
                print("doom returned", l)
            return int (l) == 1
        return False
```

isvisible

$HOME/Sandpit/git-doom3/pybot-dhewm3/neo/game/ai/pybot.h

```
    void rpcGetEntityName (char *data);
    void rpcCanSeeEntity (char *data);
    void rpcMapToRunTimeEntity (char *data);
```

isvisible

- `$HOME/Sandpit/git-doom3/pybot-dhewm3/neo/game/ai/pybot.cpp`

```
/*
 *  rpcCanSeeEntity - returns 1 if this bot can see entity, entNo,
 *                    returns 0 otherwise.
 */

void pyBotClass::rpcCanSeeEntity (char *data)
{
  if (protocol_debugging)
    gameLocal.Printf ("rpcCanSeeEntity (%s) call by python0, data);

  char buf[1024];
  int entNo = atoi (data);
  int result = 0;

  if (entNo >= 0)
    result = dictionary->canSeeEntity (rpcId, entNo);
  idStr::snPrintf (buf, sizeof (buf), "%d0, result);
  if (protocol_debugging)
    gameLocal.Printf ("rpcCanSeeEntity responding with: %s0, buf);
  buffer.pyput (buf);
  state = toWrite;
}
```

isvisible

$HOME/Sandpit/git-doom3/pybot-dhewm3/neo/game/ai/pybot.cpp

```
/*
 *  canSeeEntity - returns true if bot id can see entity entNo.
 */

bool dict::canSeeEntity (int id, int entNo)
{
  return entry[id]->canSeeEntity (entNo);
}
```

isvisible

$HOME/Sandpit/git-doom3/pybot-dhewm3/neo/game/ai/pybot.cpp

```
bool dict::aim (int id, int enemy)
{
  if (enemy <= maxId)
    return entry[id]->aim (entry[enemy]->getIdEntity ());
  else
    return entry[id]->aim (gameLocal.GetEntity (enemy));
}
```

isvisible

- `$HOME/Sandpit/git-doom3/pybot-dhewm3/neo/game/ai/pybot.cpp`

```
void pyBotClass::rpcGetPos (char *data)
{
  if (protocol_debugging)
    gameLocal.Printf ("rpcGetPos (%s) call by python\n", data);

  char buf[1024];
  int id = checkId (atoi (data));

  if (id > 0)
    {
      if (id <= maxId)
      {
        /* A pybot or human player.  */
        const idVec3 &org = dictionary->getPos (id);
        idStr::snPrintf (buf, sizeof (buf), "%g %g %g\n",
                    org.x, org.y, org.z);
      }
      else
      if (id < gameLocal.num_entities)
        {
          /* A doom3 entity.  */
          const idVec3 &org = gameLocal.entities[id]->GetPhysics ()->GetOrigin ();
          idStr::snPrintf (buf, sizeof (buf), "%g %g %g\n",
                      org.x, org.y, org.z);
        }
           else
        strcpy (buf, "None0);
    }
  else
    strcpy (buf, "None0);
  if (protocol_debugging)
    gameLocal.Printf ("rpcGetPos responding with: %s\n", buf);
  buffer.pyput (buf);
  state = toWrite;
}
```

isvisible

- notice that `maxId` is used to determine whether the entity belongs to a player/python bot
- it needs to be declared inside
- **$HOME/Sandpit/git-doom3/pybot-dhewm3/neo/game/ai/pybot.cpp**

```
static unsigned int maxId = 0;
```

isvisible

- **$HOME/Sandpit/git-doom3/pybot-dhewm3/neo/game/ai/pybot.cpp:doRegisterName**

```
...
  active.include (b, instance);
  allbots.include (b, instance);
  b->setInstanceId (instance);
  b->setRpcId (rid);
  if (rid > maxId)
    maxId = rid;
  return b;
}
```

interpretRemoteProcedureCall

$HOME/Sandpit/git-doom3/pybot-dhewm3/neo/game/ai/pybot.cpp

```
...
  else if (idStr::Cmpn (data, "can_see_entity ", 15) == 0)
     rpcCanSeeEntity (&data[15]);
  else if (idStr::Cmpn (data, "map_to_runtime_entity ", 21) == 0)
     rpcMapToRunTimeEntity (&data[21]);
...
```

Conclusion

- at this point the bot should be able to lookup a label and test whether it is visible
- the API could be cleaned up and retrofitted throughout the botlib and botcache