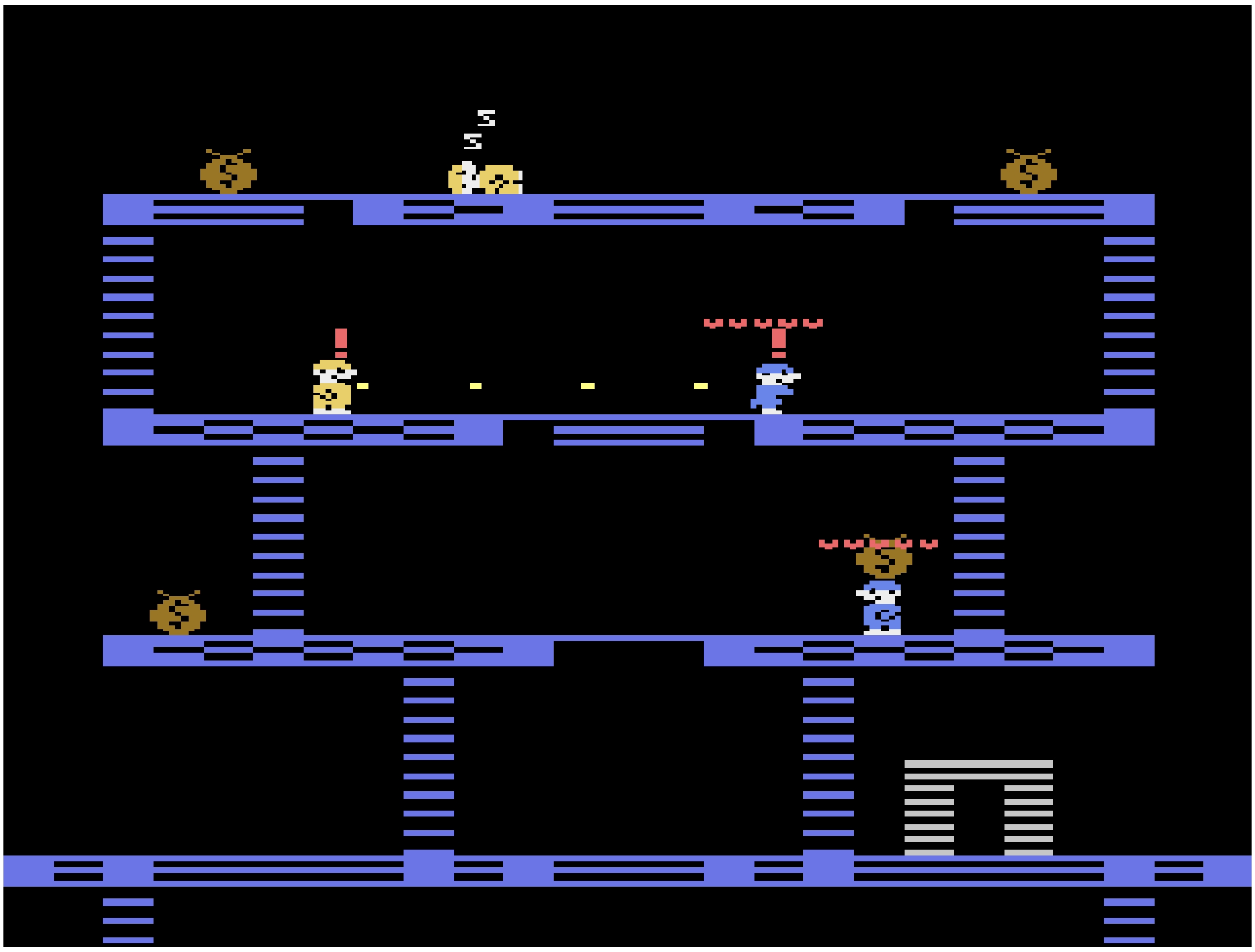
BANK ROB

Custom AI Realisation (original plan)

Milton Plotkin 9525114

HIT3046 Artificial Intelligence for Games



Concept art derived from altering multiple snapshots of Spike 7. Final product will likely have more complex navigational options.

Game Design:

The game will be an expansion on Spike 7's “patrolling guard” concept, and will feature one or more guards patrolling a bank, while thieves attempt to ambush and steal the money they protect. If shot enough times, a thief will die, but the thieves and guards will communicate information amongst each other to ensure their goal is reached.

Features:

- The guards will dynamically create a route for themselves which visits each bag of money on the field. The patrol route will change for each individual guard if they notice its absence or change of location.
- Guards fall asleep after a large amount of time without action.
- Thieves will use strategy. For example, if in plain sight, they will act casual, so as to not draw unwanted attention. Or, if caught, they'll drop what they hold to run faster.
- Use of sight and sound (or “smelly goal”) for each agent to asses its situation. This will include communication between agents to allow for teamwork.
- Breadth-First Search algorithms for movement.
- Customisable destination for thieves to escape to.

Architecture:

Note: This will be an Object Oriented game created using the SDL library with C++.

CLASSES:

Game

Contains and initialises vectors (lists) for the agents, objects and all things essential for the game. Runs all update() functions. Does not include initialisation of SDL or the main game window.

Message

Contains references to all agents and objects within the world. Sends messages to agents to inform them of any other agents they see/hear. Also handles interaction between agents and objects, such as bullet collision and bag theft.

Agent

Superclass of both Guard and Thief. Handles all actions common to both, such as movement, sight/hearing and message interpretation. Also handles a limited FSM which applies to both kinds of agent. Will also have the draw function for the agent.

Guard

Subclass of Agent. Has four main “modes” of *patrol*, *pursuit*, *investigate* and *sleep*. Will react to observing the actions of other guards or thieves, such as waking a sleeping guard or joining in a pursuit. Highest priority is to shoot any thief caught in the act of stealing.

Thief

Subclass of Agent. Has four main modes: *seek* (the money), *steal*, *casual* and *flee*. It will seek out the nearest money bag, then proceed to steal it if it won't be caught. Highest priority is self preservation, and it will not help out fellow thieves.

Bullet

Draws itself. Stores its own movement vector. Receives messages from Message to know when it hits something. Created by Guards in *pursuit*.

Money

Draws itself. Will remain motionless unless it receives a Message that it is carried/removed from the game. If its thief is killed, will drop and serve as a new waypoint for the guards once their *pursuit* is over.

Door

Draws itself. Serves as the exit location for *flee*ing and *steal*ing thieves. Location can be changed manually.

<u>Finite State Machines:</u>
**Not real code*

Agent

```
update()
{
    if dead:
        return

    if health <= 0:
        dead = sDie()

    fsm() //Virtual function (it's defined by each different Agent subclass)

    move()
    draw()
}
```

Guard

```
fsm()
{

    //Assume that once a condition for a function is met, the conditions following it are ignored
    //and not performed if it ends with return.
    //Ergo, highest priority behaviour comes first (but dying is already handles by Agent
    //superclass)

    If mode != sleep:             //If a guard is sleeping, it can't see.
    {
        sight = 20
    }

    If mode == pursuit:
    {
        If target.dead:               //Resume your duties.
        {
            mode = patrol
            return
        }

            moveToPos(target.pos)       // 'target' is a thief the guard catches stealing

        if can see target:
            shoot()

        return
    }
    If see thief with bag:
    {
        mode = pursuit
        target = thief with bag
        return
}

If hear guard in pursuit:       //An agent creates sound when its moving. This is like the
                                //guard “telling” the other about the culprit.
{
        mode = pursuit
        target = guards target
        return
}

If hear gunshot **or** see guard in pursuit:    //Note: The mode change will wake a
                                                sleeping //guard.
{
        mode = investigate
        moveToPos(guards/gunshot location)
        return
}

If investigate **and** destReached:             //For false alarms
{
        mode = patrol
        //Note, does not return.
}

If mode == patrol:
{
        If bagReached:          //Cycle through all money bags
        {
                if no bag at destination:       //Investigate missing bags
                {
                        moveToPos(exit.pos)
                        mode = investigate
                        return
                }

                        destination = nextMoneyBagLocation()
        }

        moveToPos(destination)

        //Note, does not return.
}

If see sleeping guard:
{
                Other guards mode = patrol
                return
}
```

```
If rand()%100 and mode == patrol:              //Randomly fall asleep
{
                mode = sleep
                sight = 0
                return
}
```

Thief

```
fsm()
{
        If mode != steal
        {
                speed = SPEED_NORMAL
        }

        If mode == flee and holding money:        //Money slows you down. Drop it, then run.
        {
                dropMoney()
                return
        }

        If mode == flee                           //Run until you get to the exit. Stop for nothing
        {
                moveToPos(exit.pos)
                return
        }

        If get hit:
        {
                mode = flee
                return
        }

        If mode == casual:
        {
                If holding money:
                        dropMoney()

                moveToPos(Point(self.x + 1-rand()%3, self.y)) //Wander aimlessly to look innocent
                return
        }

        If hear gunshot or hear guard or see fleeing thief:
        {
                mode = casual
                return
        }

        ///If you got here, it's safe to steal the bag
        If mode == seek
        {
                moveToPos(getNearestBag())

                if pos = getNearestBag().pos:
                        mode = steal

                return
        }
```

```
If mode == steal
{
        takeBag()
        speed = SPEED_SLOW

        moveToPos(exit.pos)

        //Note, does not return
}

if self.pos == exit.pos
{
        delete your bag
        delete yourself
}
```