1. A high-level description of each of your public member functions in each of your classes, and why you chose to define each member function in its host class; also explain why (or why not) you decided to make each function virtual or pure virtual.

class StudentWorld;

class Object : public GraphObject {

virtual void doSomething() = 0;

Pure virtual because every object does something different.

StudentWorld\* getWorld();

Returns the world the object is in.

bool isAlive() const;

Checks if the object is alive.

void setDead();

Sets the object to dead.

virtual bool canActorsPassThroughMe() const;

Differentiates between objects like boulders/dirt and squirt, some of which can be passed through and others that can’t.

bool moveToIfPossible(int x, int y);

Moves to the target location if possible

};

class Agent : public Object {

virtual void addGold() = 0;

Pure virtual because different agents react differently when picking up gold. There is no common theme.

unsigned int getHitPoints() const;

Accesses health.

virtual bool annoy(unsigned int amount);

Makes agents take damage.

};

class FrackMan : public Agent {

virtual void doSomething();

Executes all of Frackman’s moves. Moves, squirts water, uses sonar, and drops nuggets.

virtual bool annoy(unsigned int amount);

Makes frackman take damage.

virtual void addGold();

Gives Frackman one gold piece.

void addSonar();

Gives Frackman one sonar equipment piece.

void addWater();

Gives Frackman 5 squirts of water.

unsigned int getGold() const;

Accesses Frackman’s gold inventory.

unsigned int getSonar() const;

Accesses Frackman’s sonar count.

unsigned int getWater() const;

Accesses Frackman’s squirt count.

};

class Protester : public Agent {

virtual void doSomething() = 0;

Pure virtual because this function does nothing.

virtual bool annoy(unsigned int amount) = 0;

Pure virtual because it doesn’t do anything.

virtual void addGold();

Gives gold to protester.

virtual bool huntsFrackMan() const;

Sets state to hunting Frackman since the protester is close.

void setMustLeaveOilField();

If has been bribed.

bool isLeave();

Accesses the state of protester; whether or not he/she has been bribed.

};

class RegularProtester : public Protester {

virtual void doSomething();

If bribed leave in the top right corner. Follow nearby Frackman. If sees Frackman from far away turn in the direction.

virtual void addGold();

Pockets gold from bribed nuggets. Changes state.

};

class HardcoreProtester : public Protester {

virtual void doSomething();

virtual void addGold();

};

class Dirt : public Object {

virtual void doSomething();

Does nothing because it’s dirt.

};

class Boulder : public Object {

virtual void doSomething();

3 States: stable, waiting, and falling. If it hits something, delete it.

virtual bool canActorsPassThroughMe() const;

};

class Squirt : public Object {

virtual void doSomething();

Annoys protesters by traveling out from Frackman a short distance.

};

class ActivatingObject : public Object {

virtual void doSomething();

Includes all of the pickupable objects. Doesn’t do anything.

bool isVisible();

All objects of this class need to access visibility.

bool discovered();

Works for the objects in this class. Sets the visibility if has been passed by Frackman.

};

class OilBarrel : public ActivatingObject {

virtual void doSomething();

If within radius of Frackman, it is discovered. Sends notification to studentworld to keep track of barrels since they decide when the level ends.

};

class GoldNugget : public ActivatingObject {

virtual void doSomething();

This is the object pickupable by Frackman. Most functionality is described in parent class Activating Object.

};

class BribeNugget : public ActivatingObject {

virtual void doSomething();

};

class SonarKit : public ActivatingObject{

virtual void doSomething();

Reveals area surrounding Frackman.

};

class WaterPool : public ActivatingObject{

virtual void doSomething();

A lot of functionality described in parent class Activating Object. Main difference is point value assigned when picked up.

};

2. A list of all functionality that you failed to finish as well as known bugs in your classes, e.g. “I wasn’t able to implement the Squirt class.” or “My Hardcore Protester doesn’t work correctly yet so I just treat it like a Regular Protester right now.”

Protesters don’t work. I implemented the most of the member functions but because I was running out of time I didn’t program incrementally. When I add protesters during init() in studentworld, the program crashes. Therefore, bribenuggets (gold nuggets that protesters don’t work either). Also water pools don’t appear on the map in the random fashion that they’re supposed to.

3. A list of other design decisions and assumptions you made.

Instead of a single gold nugget class, I split it into two to make things easier. One of the gold nuggets is the one that Frackman picks up, the other is dropped by Frackman and picked up by protesters. They both use the same image ID.

4. A description of how you tested each of your classes (1-2 paragraphs per class).

Frackman

Walked to each corner. Walked to each boulder and tried different positions underneath it. Used all the functions associated with Frackman: squirt, sonar, dropping bribe nuggets. Set sonar to radius 40 to make checking everything easier.

Dirt

Doesn’t really do anything.

Boulder

Falls on stuff. Checked the falling by inching underneath it w/ Frackman. Spawned the boulders manually in different places to make sure it didn’t fall prematurely.

Squirt

Tried squirting on top floor/bottom floor/edge of map. Tried to squirt through walls and through boulders as well as the other objects in the game. Gave myself infinite squirts to facilitate the process.

Barrel

Set sonar’s radius to 40 to make testing easier. Spawned multiple barrels manually to check the object array. Collected barrels.

Nugget

Increased sonar’s radius as previously explained. Spawned them on top on another. Collected them like a maniac.

Bribe Nugget

Sonar

Set radius to the entire map. Gave myself unlimited sonar charges because finding things sucks. Spawned at a rate of one per tick since I don’t like waiting.

Water Pool

Spawned them in the corner in one place at a rate of one per tick to refill and test my squirts.