A.

**Classes Relationships:**

GameWorld:

Derived: StudentWorld

GraphObject:

Derived: Unit

Derived: Frackman

Derived: Protestor

Derived: InvisibleItem

Derived: Oil

Gold

Sonar

WaterPool

Squirt

Boulder

Dirt

**Class descriptions:**

1. StudentWorld has a vector to contain everything except Dirt and FrackMan, Dirt pointers are stored in the 2D array to access easily. Frackman is another pointer.
2. Each derived classes from Unit would have a pointer to the StudentMap to easily call the methods of StudentWorld
3. In order to avoid using Class ID to identify object type, several virtual functions, such as:

virtual bool AreYouBoulder();

virtual bool AreYouProtector();

were used. They were overwritten in the real derived class to determine whether this is the class type.

1. InvisibleItem is the base class for every classes except Protestor.

It has two methods, bool IsDetected(FrackMan\* p);

bool IsConsumed(FrackMan\* p);

These two functions are used to monitor the distance between the InvisibleItem and FrackMan

1. For the interactions between Protestor and derived classes of InvisibleItem, the functions in the StudentWorld were used. Such as:

bool StudentWorld::harmProtestor(int X, int Y, int r,int harm)

bool StudentWorld::bribeProtestor(int X, int Y, int r)

Those functions would traverse the vector and interacts with the corresponding classes.

1. IndexWorld function in the StudentWorld in the class is used for the best path routing algorithms in the Protestor leaving field function and searching function. It implemented Lee’s routing algorithm and construct a map with self defined coordinate struct. The coordinate struct consists of current coordinate and the next step in the best rout under current conditions.
2. getCoord\_nextX(int current\_X,int current\_Y) and getCoord\_nextY(int current\_X,int current\_Y) would return the next step in the best route to the destination.
3. Protestor use sitIntersection() to determine whether they are in an intersection and use turnToFrackman() to move closer to Frackman.
4. The rest of functions are quite trivial and they were used to detect the surrounding environment, such as IsDirtBeneath() or IsSquirtBloc() were respectively used to detect whether there are dirts below this object and whether squirt is blocked by dirt or boulder.

**B.**

I failed to realized the Hardcore protestor. (But the best routing algorithms has been implemented in the follow function in the comments)

**C.**

I make Player move first before all the computer-controlled object to obtain a better experience for user.

**D.**

The test case were Player and Protestor centralized.

Player.

Dig dirts (Dig to the boundary, dig to let the boulder fall....)

Pickup Gold

Drop gold

Squirt Water

Pickup Water

Use Sonar

Pickup Sonar

Protester:

Pickup Gold and leave field, make sure its route is best

Be annoyed and leave field, make sure its route is best

Walk “randomly” before seeing Frackman or within a distance to follow him

Use Frackman to create a maze to see if protester can work out.