GradientDecentOptimizer

```
    Mesh* _mesh;
    DeviceMesh* _DMesh;
    Gradient* _gradient;
    DeviceAPI* _GPU;
    double tol = 1e-10;
    double _startingVol = 0;
    double _stepSize = 0.1;
    double _dAtol = 1e-8;
    int _maxConstraintSteps = 20;

    double gradientDesentStep(); // takes a gradient step
    double reproject_constraints();
```

```
    ShapeOptimizer(const char * fileName);
    ~ShapeOptimizer();

    double gradientDesent(int); // do n steps
    int optimize();
    void printMesh(const char*);
    Mesh get_optimized_mesh(){return _DMesh->copy_to_host();}
```

Mesh

```
unsigned int _numVert
double* _vert
unsigned int _numFacet
unsigned int * _facet
```

```
Mesh(const char *fileName)
print(const char *fileName)
updateFromGradient(Gradeint
```

DeviceMesh

```
   unsigned int _numVert = 0;
 unsigned int _numFacets = 0;

 DeviceAPI * _GPU = nullptr;


 UniqueDevicePtr<double> _vert = UniqueDevicePtr<double>(_GPU);
 UniqueDevicePtr<unsigned> _facets = UniqueDevicePtr<unsigned>(_GPU);

 // arrays holding the map from vertex to <facet, # in facet>
 UniqueDevicePtr<unsigned> _vertToFacet = UniqueDevicePtr<unsigned>(_GPU); // the a list of facet i
 UniqueDevicePtr<unsigned> _vertIndexStart = UniqueDevicePtr<unsigned>(_GPU);// where the indcies i

 UniqueDevicePtr<double> _area = UniqueDevicePtr<double>(_GPU);// holds the area per facet
 UniqueDevicePtr<double> _volume = UniqueDevicePtr<double>(_GPU);// holds the volume per facet
```

```
DeviceMesh(const char *fileName)
DeviceMesh(Mesh) // copies a mesh to the GPU

Mesh copyToHost() // copies a mesh from the GPU

updateFromGradient() //calls moveMesh kernal
```

Gradient

```
double *_gradA
double *_gradV
double *_gradAProjected

DeviceAPI * _GPU = nullptr;
```

```
-- Mesh methods are CPU and for time comparison
Gradient(Mesh) // uses CPU memory, sets _onGPU = fal
Gradient(DeviceMesh) // uses GPU memory sets _onGPU =

calculateGradA(Mesh)
calculateGradA(DeviceMesh) // calls calcA kernal
calculateGradV(Mesh)
calculateGradV(DeviceMesh) // calls calcV kernal
projectGradA(Mesh)
projectGradA(DeviceMesh)   // calls projectA Kernal
```

UniqueDevicePtr<T>

```
    void* _value = nullptr;
    DeviceAPI* _myDevice;
```

```
UniqueDevicePtr(DeviceAPI* apiIn);
UniqueDevicePtr(void* ptrIn, DeviceAPI* apiIn);

~UniqueDevicePtr(){if(_value) _myDevice->deallocate(_value);}
void allocate(int size)
void* get_void(){return _value;}
T* get(){return (T*) _value;}
```

Device_API

```
unsigned int _blockSize
```

```
    virtual void allocate(...) = 0;
    virtual void copy_to_host(...) = 0;
    virtual void copy_to_device(...) = 0;
    virtual void deallocate(...) = 0;
    virtual double getGPUElement(...) = 0;
    virtual unsigned int getGPUElement(...) = 0;


    virtual double sum_of_elements(...) = 0;
    virtual double dotProduct(...) = 0;
    virtual void add_with_mult(...) = 0;//a = a + b* lambda
    virtual void project_force(...) = 0;
    virtual void facet_to_vertex(...) = 0;
    virtual void area_gradient(...) = 0;
    virtual void volume_gradient(...) = 0;
    virtual void area(...) = 0;
    virtual void volume(...) = 0;

    DeviceAPI(unsigned int blockSizeIn){blockSize = blockSizeIn;}

    unsigned int get_blockSize(){return blockSize;}
    void set_blockSize(unsigned int size){ blockSize = size;}
```