FUNCTIONALITY :

Classes that hold none or little functionality, have entire data is made up of primitive types, and are less than 64 bytes can be made into structs that are passed and taken directly as data.

Any other classes will be turned into partial classes with one file of the class name declaring it as a child of ‘BufferType’ or a derived type of ‘BufferType’. The other class with the name of class name plus ‘Include’ will give the class functions to retrieve the variables it needs.

BufferType and derived types will be initialized with IntPtr to the location of the requested class type.

Classes that hold other pointers to other classes will return an initialized class with the pointer

Classes that hold other classes will use a private internal value initialized in the constructor. The public value will return the internal one for gets, and the set will go through an external function

Classes that hold other structs will

Array type will be used in replacement of spans, arrays, array pointers, or any other array types that hold constant contiguous memory. Their lengths are immutable but do allow to change the data inside the array. It is an exception to attempt to retrieve or set data past the length of the array.
Each type that uses an array will need to define functions for when (if) it is possible to change the array length.

Vectors that hold pointers to a type will use the ‘PtrVector’ type and its functions. It should be initialized in the constructor and have a private set.

FORMATING :
Comment line is a comment with 50 dashes //--------------------------------------------------

Every type within a class and (if it has) its helper variables should be enclosed in comment lines with a comment on the line above the first comment. It should be in parentheses the type and then name, as if using a c-style cast. For Object mesh it would be
//(Mesh)mesh
For grouped items it should be the main item name and type
For private public combo, they should be grouped within the same comment lines and the private variable on the line directly above the public. The comment should describe the public variable,

Class and struct first constructors/destructors should have 5 spaces from the closet non-empty line, except if the constructor/destructor is the only thing within the class
The next (if any) constructors/destructors should have one space empty between each other

There should be 3 spaces between the types in a class, except when it is the very first one
Struct variables don't need comment lines or spaces, but they should be used if needed.

In all the 'Include' classes the groups should be separated with a single empty line, and relative functions and variables placed on lines directly above or below each other.
For larger 'Include' classes the use of comment lines and more spaces to separate groups should be used.

Class and struct functions should have a single top comment line and description comment above that
Their description comment should have 3 spaces from the nearest non-empty line, except for the first which should have 5 spaces away

Class and struct operators should all be enclosed within comment lines and each one should have a comment above with an example use. The main top comment line should be 5 spaces away from the nearest non-empty line. The type of each thing in the example should be within parenthesis before the type as if performing a c-style cast. For Material implicit operator with an IntPtr the comment would be
//(Material)mat = (IntPtr)4
Their top comment lines should have 3 spaces from the nearest non-empty line
Each operator example comment should be 2 spaces away from the nearest non-empty line, except for the first one.

The order things will go in within classes and structs is :

1. Variables
2. Functions
3. Operators
4. Constructors
5. Destructors

When there are multiple classes in a file, each one should be enclosed in comment lines.
Each class should have 5 spaces from the nearest closet non-empty line, except for the starting class