References vs Pointers

Pointers and references can have similar use cases in C++. As seen previously both references and pointers can be used in pass-by-reference to a function. Additionally, they both provide an alternative way to access an existing variable: pointers through the variable's address, and references through another name for that variable. But what are the differences between the two, and when should each be used? The following list summarizes some of the differences between pointers and references, as well as when each should be used:

References	Pointers
References must be initialized when they are declared. This means that a reference will always point to data that was intentionally assigned to it.	Pointers can be declared without being initialized, which is dangerous. If this happens mistakenly, the pointer could be pointing to an arbitrary address in memory, and the data associated with that address could be meaningless, leading to undefined behavior and difficult-to-resolve bugs.
References can not be null. This means that a reference should point to meaningful data in the program.	Pointers can be null. In fact, if a pointer is not initialized immediately, it is often best practice to initialize to `nullptr`, a special type which indicates that the pointer is null.
When used in a function for pass-by-reference, the reference can be used just as a variable of the same type would be.	When used in a function for pass-by-reference, a pointer must be dereferenced in order to access the underlying object.

References are generally easier and safer than pointers. As a decent rule of thumb, references should be used in place of pointers when possible.

However, there are times when it is not possible to use references. One example is object initialization. You might like one object to store a reference to another object. However, if