SOLID Principles Julio Echavarría

Estos principios se usan para que el código sea más flexible y fácil de usar.

- **Single Responsibility principle**
  Este principio se refiere a que cada método que creemos se debe ocupar de solo una cosa, si deseamos hacer varias cosas en una sola, debemos separarlas y luego ejecutar dichos métodos para que hagan lo que deseábamos.
  En el video se puede ver como separa cada responsabilidad (crear el rayo, determinar si se seleccionó, cambiar la textura o activar overlays, etc.)

- **Open-Closed principle**
  Las clases deben estar abiertas a extensiones y cerradas a modificaciones.
  En el video, se creó una clase que utiliza el como se comporta la clase que selecciona y deselecciona, y se le cambió el comportamiento para que utilizara un asset del store de Unity, de esta forma se modifica lo que hace sin perder el comportamiento anterior.

- **Liskov Substitution principle**
  Los objetos de una clase deben poder ser reemplazados por objetos de sus subclases sin que esto rompa el comportamiento o funcionamiento de la aplicación.
  En el video esto se da reemplazando la implementación principal de el highlight por una interfaz.

- **Interface Segregation principle**
  Ningún código debe ser forzado a depender de un método que no utilice, por lo tanto, es mejor tener varias interfaces que una grande.
  En el video se dividió la respuesta al mirar en diferentes partes, así no es necesario que utilice todo el código para funcionar.

- **Dependency Inversion principle**
  Todos los módulos deben depender de abstracciones y no de distintos módulos para funcionar.
  En el video se estaba referenciando una implementación concreta, después de aplicar el principio, mira una abstracción en la que no sabe como va a responder, solo sabe que necesita delegar información a otra parte del código.