MinilibX Kütüphanesi : MinilibX, özellikle basit grafiksel uygulamaların (örneğin, oyunlar veya görselleştirmeler) oluşturulması için kullanılan bir grafik kütüphanesidir.

Mlx_hook fonksiyonu : Olay işleme (event handling) için kullanılır. Grafik uygulamalarında, fare hareketleri, klavye girişleri veya pencere olayları gibi çeşitli olaylara tepki verme ihtiyacı doğabilir. mlx_hook, bu tür olaylara özel bir fonksiyon atanmasını sağlar.
Genel kullanım şu şekildedir:

```
int mlx_hook(void *win_ptr, int event_type, int (*funct_ptr)(), void *param);
```

1. win_ptr: Olayların bağlı olduğu pencereye (window) işaret eden bir pointer.
2. event_type: Hangi olayın işleneceğini belirten bir tamsayı değeri. Örneğin, fare tıklamalarını veya klavye tuşlarına basılmalarını temsil eden özel değerler olabilir.
3. funct_ptr: Belirli bir olay gerçekleştiğinde çağrılacak işlevin işaretçisi (pointer).
4. param: İşlevin çağrılması sırasında kullanılacak ek parametrelerin bir işaretçisi.

Bu fonksiyon, belirli bir olay gerçekleştiğinde çağrılacak olan işlevi belirlemenizi sağlar. Örneğin, bir fare tıklaması durumunda bir işlev çağrılabilir ve bu işlev tarafından belirli bir tepki oluşturulabilir. Bu, grafik uygulamalarda kullanıcı etkileşimlerine yanıt vermek için yaygın bir tekniktir.

olay işleme (event handling) nedir? : Olay işleme (event handling), bir bilgisayar programında gerçekleşen olaylara (events) tepki verme sürecidir. Bir olay, genellikle kullanıcının bir etkileşimde bulunduğu bir durumu temsil eder. Bu etkileşimler, fare tıklamaları, klavye tuşlarına basmalar, pencere boyutu değişiklikleri, ağ olayları, zamanla ilgili olaylar ve daha birçok şeyi içerebilir.
Olay işleme genellikle şu adımları içerir:

1. Olayın Algılanması (Event Detection): Kullanıcının bir etkileşimde bulunması sonucu ortaya çıkan olaylar, program tarafından algılanır. Bu, kullanıcının bir tuşa basması, fareyi hareket ettirmesi veya benzeri etkileşimler olabilir.
2. Olayın Tanımlanması (Event Identification): Algılanan olay, ne tür bir olay olduğunu belirten bir tür veya kod ile tanımlanır. Örneğin, fare tıklaması bir tür kod ile temsil edilebilir.
3. Olayın İşlenmesi (Event Handling): Tanımlanan olaya yanıt olarak belirli bir işlev veya işlev seti çağrılır. Bu işlevler, olayın program içinde nasıl işleneceğini belirler.
4. Gerekirse Olaya Yanıt Verme (Event Response): İşlenen olaya bağlı olarak program, gerekirse kullanıcıya bir yanıt gönderebilir. Örneğin, bir tuşa basıldığında ekranda bir metin değişikliği yapılabilir.

Event handling, özellikle kullanıcı arayüzü (UI) tasarımı ve grafik uygulamalar gibi interaktif uygulamalarda önemli bir konsepttir. Modern programlama dilleri ve kütüphaneleri, olay işleme için genellikle özel araçlar ve yöntemler sunar. Şimdi

So_long örneğine bakalım.

```
mlx_hook(data.winpointer, 17, 0, (void *)exit, 0);
```

1. data.winpointer: Bu, olayların bağlı olduğu pencereye işaret eden bir pointer'dır. Kodun geri kalanında bu pencereye data.winpointer ile erişilecek.
2. 17: Bu, işlenecek olayın türünü belirten bir tamsayıdır. Burada 17, pencerenin kapatılma olayını temsil eder. 17 numarası, pencerenin "Close Window" (Kapat) düğmesine tıklanması olayını ifade eder. Bu olay türü, genellikle programın düzgün bir şekilde sonlandırılmasını sağlamak için kullanılır.

3. 0: Bu, bir işlevin döndüreceği değeri belirten bir tamsayıdır. Bu örnekte, işlevin herhangi bir değer döndürmediği durumu temsil eder, yani 0 olarak belirtilmiştir.
4. (void *)exit: Bu, belirli bir olay gerçekleştiğinde çağrılacak olan işlevin işaretçisidir. Burada exit fonksiyonuna bir işaretçi atanmıştır. Yani, pencere kapatma olayı meydana geldiğinde program exit fonksiyonunu çağıracaktır. Bu, programın kapatılmasını sağlar.
5. 0: Bu, işlevin çağrılması sırasında kullanılacak ek parametrelerin bir işaretçisidir. Bu örnekte, ek bir parametre kullanılmadığı için 0 olarak belirtilmiştir.

Bu kod satırı, özellikle pencerenin kapatılma olayına yanıt olarak programın düzgün bir şekilde sonlandırılmasını sağlamak üzere kullanılmıştır. Yani, kullanıcı pencereyi kapattığında, program exit fonksiyonunu çağırarak sonlanacaktır.

mlx_hook(data.winpointer, 2, 0, &controls_working, &data);

1. data.winpointer: Bu, data yapısındaki bir öğedir ve pencerenin (window) işaretçisini temsil eder. Diğer bir deyişle, bu, mlx_hook fonksiyonuna bağlı olacak pencerenin işaretçisidir.
2. 2: Bu, işlenecek olayın türünü belirten bir tamsayı değeridir. Burada 2, klavye tuşlarına basılma olayını temsil eder. Özellikle, klavyeden bir tuşa basıldığında bu olay tetiklenecektir. (W,A,S,D)
3. 0: üçüncü parametre, olayın nasıl işleneceğini kontrol eden bir bayrak değeri olup, bu değeri kontrol etmek 0 olarak belirlenmiş Bu durumda, 0 olarak belirlenmiş olması, özel bir işlemin uygulanmadığını ve olayın varsayılan işlemi görmesi gerektiğini ifade eder. Eğer farklı bir değer kullanılsaydı, olayın özel bir şekilde işlenmesi, engellenmesi veya değiştirilmesi anlamına gelebilirdi.
4. &controls_working: Bu, belirli bir olay gerçekleştiğinde çağrılacak olan işlevin işaretçisidir. Burada controls_working adlı bir işlev belirlenmiştir. Bu işlev, klavye tuşlarına basılma olayına yanıt verecek şekilde tasarlanmıştır. (muhtemelen bunun içinde klavye tuşlarının sayıları var)
5. &data: Bu, işlevin çağrılması sırasında kullanılacak ek parametrelerin bir işaretçisidir. Bu örnekte, işlev t_data türünden bir yapıyı kullanacak ve bu yapı içinde gerekli verilere erişim sağlanacaktır.

Bu satır, klavyeden bir tuşa basılma olayını dinlemek ve bu olaya yanıt olarak controls_working işlevini çağırmak üzere kullanılır. controls_working işlevi, klavyeden tuşa basılma olayına yanıt veren bir işlevdir ve bu örnekte pencereyi kapatma işlemini gerçekleştirir.

Mlx_loop fonksiyonu: mlx_loop, MinilibX kütüphanesinde grafik uygulamalarında kullanılan bir fonksiyondur. Bu fonksiyon, bir grafik penceresinin olay döngüsünü başlatmak ve uygulamanın sürekli olarak çalışmasını sağlamak için kullanılır.

genel kullanım:

int mlx_loop(void *mlx_ptr);

1. mlx_ptr: MinilibX bağlamını temsil eden bir işaretçidir. Bu işaretçi, MinilibX kütüphanesini başlatmak için kullanılır. (So_long'da da aynı.)

Bu fonksiyon, programın grafiksel kullanıcı arayüzünü sürekli olarak güncel tutmak ve olaylara anında tepki verebilmek için kullanılır. mlx_loop çağrıldıktan sonra, program sürekli olarak olayları bekler ve gelen olaylara göre işlem yapar. (kısaca sürekli çalışmasını sağlar.)

Hazırlayan : Derya Acar