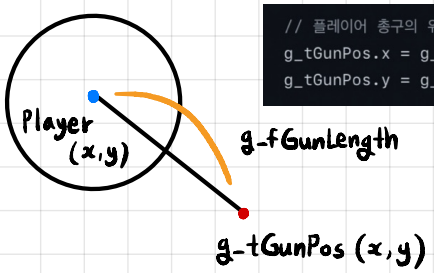
08. 삼각함수 활용



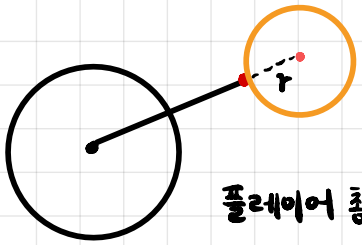

```
// 플레이어 총구의 위치를 구해준다.
g_tGunPos.x = g_tPlayer.x + cosf( g_fPlayerAngle ) * g_fGunLength; // cosf 값은 0~PI 기준.
g_tGunPos.y = g_tPlayer.y + sinf( g_fPlayerAngle ) * g_fGunLength;
```

```
if( GetAsyncKeyState( 'D' ) & 0x8000 )
{
    g_fPlayerAngle += PI * g_fDeltaTime * fTimeScale;
}
if( GetAsyncKeyState( 'A' ) & 0x8000 )
{
    g_fPlayerAngle -= PI * g_fDeltaTime * fTimeScale;
}
if( GetAsyncKeyState( 'W' ) & 0x8000 )
{
    g_tPlayer.x += fSpeed * cosf( g_fPlayerAngle );
    g_tPlayer.y += fSpeed * sinf( g_fPlayerAngle );
}
if( GetAsyncKeyState( 'S' ) & 0x8000 )
{
    g_tPlayer.x -= fSpeed * cosf( g_fPlayerAngle );
    g_tPlayer.y -= fSpeed * sinf( g_fPlayerAngle );
}

// 총구 위치 구하기
g_tGunPos.x = g_tPlayer.x + cosf( g_fPlayerAngle ) * g_fGunLength;
g_tGunPos.y = g_tPlayer.y + sinf( g_fPlayerAngle ) * g_fGunLength;
```

초당 180도 회전

총구 각도로 이동

```
/* 플레이어 총알 생성 */
if( GetAsyncKeyState( VK_SPACE ) & 0x8000 )
{
    BULLET tBullet = {};

    tBullet.circle.r = g_tPlayer.r / 2.f;
    tBullet.circle.x = g_tGunPos.x + cosf( g_fPlayerAngle ) * tBullet.circle.r;
    tBullet.circle.y = g_tGunPos.y + sinf( g_fPlayerAngle ) * tBullet.circle.r;
    tBullet.fDist = 0.f;
    tBullet.fLimitDist = 500.f;
    tBullet.fAngle = g_fPlayerAngle;

    g_PlayerBulletList.push_back( tBullet );
}
```

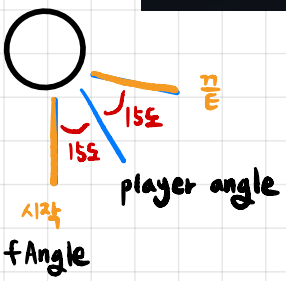


플레이어 총알 생성

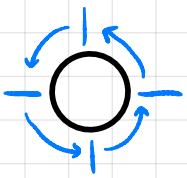
총알이동

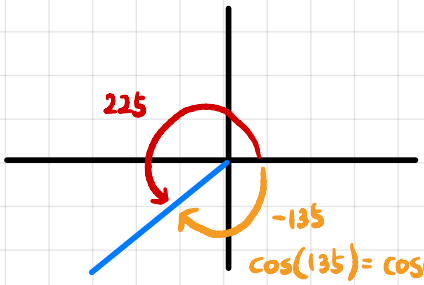
```
iter->circle.x += fSpeed * cosf(iter->fAngle) * fTimeScale;
iter->circle.y += fSpeed * sinf(iter->fAngle) * fTimeScale;
```

```
// 여러개 총알 15도
if( GetAsyncKeyState( '1' ) & 0x8000 )
{
    float fAngle = g_fPlayerAngle - PI / 12.f;

    for( int i = 0; i < 3; ++i )
    {
        BULLET tBullet = {};

        tBullet.circle.r = g_tPlayer.r / 2.f;
        tBullet.circle.x = g_tGunPos.x + cosf( fAngle ) * tBullet.circle.r;
        tBullet.circle.y = g_tGunPos.y + sinf( fAngle ) * tBullet.circle.r;
        tBullet.fDist = 0.f;
        tBullet.fLimitDist = 500.f;
        tBullet.fAngle = fAngle;

        g_PlayerBulletList.push_back( tBullet );
        fAngle += PI / 12.f;
    }
}

// 여러개 총알 사방 10도씩 36번 -> 360도
if( GetAsyncKeyState( '2' ) & 0x8000 )
{
    float fAngle = 0.f;

    for( int i = 0; i < 36; ++i )
    {
        BULLET tBullet = {};

        tBullet.circle.r = g_tPlayer.r / 2.f;
        tBullet.circle.x = g_tGunPos.x + cosf( fAngle ) * tBullet.circle.r;
        tBullet.circle.y = g_tGunPos.y + sinf( fAngle ) * tBullet.circle.r;
        tBullet.fDist = 0.f;
        tBullet.fLimitDist = 500.f;
        tBullet.fAngle = fAngle;

        g_PlayerBulletList.push_back( tBullet );
        fAngle += PI / 18.f;
    }
}
```



사방으로 10도씩 36개






```
// 각도는 오직 CCW(Counter Clock Wise) 방향으로만 회전한다고 제한.
/*
    cos(135°) = -sqrt(2)/2
    근데, cos(225°) 또한 -sqrt(2)/2임. acos 함수의 범위는 0 ~ PI 이므로,
    acos(-sqrt(2)/2)를 할 경우, 135도 값에 해당하는 라디안 값을 리턴해 줄 것임.
    따라서, 225도에 해당하는 라디안 값을 얻기 위해 다음과 같이 함.
    225 = 360 - 135 이고, target의 y 값이 origin의 y값 보다 작을 경우, 180도를 넘어간 것으로 간주.
    360도의 라디안 값 2 * PI 에서 acos 결과로 리턴된 라디안 값을 빼준다.
    즉, cos(theta) = cos(-theta) 이기 때문에, 부호 없다고 생각하고 전체 원 360 - 135 = 225를 수행하는 것.
    이렇게 해서 각도를 구해주면 비로소 135도와 225도에 해당하는 라디안 값들을 얻게 되는데,
    x좌표는 위에 언급한바와 같이 135도, 225도 둘 다 값지만,
    y좌표의 경우 sin(135°) = sin(pi - 45°) = sin45° / sin(225°) = sin(pi + 45°) = -sin45° 로 y좌표의
    부호가 서로 달라지게 되어 이를 총알의 움직임에 사용하면, 타겟이 위아래로 움직이건 간에 정확히 타겟을 향하게 된다.
*/
float GetAngle( FPOINT origin, FPOINT target )
{
    float x = (target.x - origin.x);
    float y = (target.y - origin.y);

    float dist = sqrt(x * x + y * y);

    if( target.y < origin.y )
    {
        return 2 * PI - acos( x / dist );
    }
    else return acos( x / dist );
}
```

<몬스터 총알 생성>

```
tBullet.circle.r = 25.f;
tBullet.circle.x= g_tMonster.tCircle.x - g_tMonster.tCircle.r - tBullet.circle.r;
tBullet.circle.y = g_tMonster.tCircle.y;
tBullet.fDist = 0.f;
tBullet.fLimitDist = 800.f;
tBullet.fAngle = GetAngle( FPOINT{ tBullet.circle.x, tBullet.circle.y }, FPOINT{ g_tPlayer.x, g_tPlayer.y } );

g_MonsterBulletList.push_back( tBullet );
```