```
module ring(
      outerRadius,
      innerRadius,
      thickness,
      nFrags = 100 )
{
      difference()
      {
            cylinder( h = thickness, r = outerRadius, $fn = nFrags );
            cylinder( h = thickness, r = innerRadius, $fn = nFrags );
      }
}

module ewTooth(
      ewtHeight,
      ewtTrailingAngle,
      ewtIncludedAngle,
      ewtThickness )
{
      rotate( -ewtTrailingAngle, [ 0, 0, 1 ] )
      rotate( 180, [ 0, 0, 1 ] )
      difference()
      {
            cube( [ ewtHeight, ewtHeight, ewtThickness ] );
            translate( [ 0, 0.85, 0 ] ) // Tooth tip thickness.
            rotate( ewtIncludedAngle, [ 0, 0, 1 ] )
            cube( [ ewtHeight * 2, ewtHeight, ewtThickness ] );
      }
}

module ringWithTeeth(
      ewTipRadius,
      ewRootRadius,
      ewRimWidth,
      ewToothCount,
      ewThickness )
{
      ewRimRadius = ewRootRadius - ewRimWidth - 0.75; // Hide teeth' roots.

      union()
      {
            ring( ewRootRadius, ewRimRadius, ewThickness );
            for ( t = [ 0 : ewToothCount ] )
            {
                rotate( t * EwAngularPitch, [ 0, 0, 1 ] )
                translate( [ ewTipRadius, 0, 0 ] )
                children( 0 );
            }
      }
}

module ew(
      ewTipRadius,
```

```
      ewRootRadius,
      ewRimWidth,
      ewToothCount,
      ewtTrailingAngle,
      ewtIncludedAngle,
      ewSpokesCount,
      ewHubOuterRadius,
      ewHubInnerRadius,
      ewThickness = StdThickness )
{
      ewToothHeight = ewTipRadius - ewRootRadius + ewRimWidth;
      ewSpokeHeight = ewRootRadius - ewHubInnerRadius;

      union()
      {
            /*
               The escape wheel's hub.
             */
            ring( ewHubOuterRadius, ewHubInnerRadius, ewThickness );

            /*
               The wheel itself plust the teeth.
             */
            ringWithTeeth( ewTipRadius,
                  ewRootRadius,
                  ewRimWidth,
                  ewToothCount,
                  ewThickness )
                  ewTooth( ewToothHeight,
                  ewtTrailingAngle,
                  ewtIncludedAngle,
                  ewThickness );

            /*
               The spokes.
             */
            for ( s = [ 0 : ewSpokesCount ] )
            {
                  rotate( s * EwInterSpokeAngle, [ 0, 0, 1 ] )
                  translate( [ ewHubInnerRadius, -EwSpokeWidth / 2, 0 ] )
                  cube( [ ewSpokeHeight, EwSpokeWidth, ewThickness ] );
            }
      }
}

module entryPallet()
{
    union()
    {
        cube( [ ForkOuterRadius, ForkOuterRadius, StdThickness ] );
        rotate( DnCnF, [ 0, 0, 1 ] )
        cube( [ ForkOuterRadius, ForkOuterRadius, StdThickness ] );
    }
```

```
}

module dbfork(
      fHubOuterRadius,
      fHubInnerRadius,
      fThickness = StdThickness )
{
      /*
         The vertical axis of symmetry of the fork is OX.
       */
      union()
      {
            /*
               The fork's hub.
             */
            ring( fHubOuterRadius, fHubInnerRadius, fThickness );

            /*
               The Entry pallet. The fork's arm goes first.
             */
            rotate( 90 + ForkEntryArmAngle, [ 0, 0, 1 ] )
            translate( [ fHubInnerRadius, -ForkArmWidth / 2, 0 ] )
            cube( [ FokrArmLength, ForkArmWidth, fThickness ] );

            /*
               The pallet itself.
             */
            intersection()
            {

                  ring( ForkOuterRadius, ForkInnerRadius, fThickness );
                  translate( [ -ImpFcX, ImpFcY, 0 ] )
                  rotate( -CnFW - DnCnF, [ 0, 0, 1 ] )
                  entryPallet();
                  /*
                     This cube cuts off the extra ring.
                   */
                  rotate( 90 + ForkEntryArmAngle, [ 0, 0, 1 ] )
                  cube( [ ForkOuterRadius, ForkOuterRadius, fThickness ] );
            }

            /*
               The Exit pallet. The fork's arm goes first.
             */
            rotate( -90 - ForkExitArmAngle, [ 0, 0, 1 ] )
            translate( [ fHubInnerRadius, -ForkArmWidth / 2, 0 ] )
            cube( [ FokrArmLength, ForkArmWidth, fThickness ] );

            /*
               The pallet itself.
             */
            intersection()
            {
```

```
                    ring( ForkOuterRadius, ForkInnerRadius, fThickness );
                    translate( [ -ImpFcX, -ImpFcY, 0 ] )
                    rotate( -( 90 - CnFW + DnCnF ), [ 0, 0, 1 ] )
                    cube( [ ForkOuterRadius, ForkOuterRadius, fThickness ] );
                    /*
                       This cube cuts off the extra ring.
                     */
                    rotate( 180 - ForkExitArmAngle, [ 0, 0, 1 ] )
                    cube( [ ForkOuterRadius, ForkOuterRadius, fThickness ] );
            }
        }
}
```

```
/*
   A Basic Scaling Unit for the entire drawing.
 */
BSU = 1.0;
StdThickness = 3 * BSU;
```

```
/*
   The Escape Wheel parameters.
 */
EwTipRadius = 60 * BSU;
EwRootRadius = 0.75 * EwTipRadius;
EwRimWidth = 0.085 * EwTipRadius;
EwToothCount = 30;
EwToothTrailingAngle = 18;
EwToothIncludedAngle = 12;
EwSpokesCount = 5;
EwSpokeWidth = EwRimWidth;
EwHubOuterRadius = 7 * BSU;
EwHubInnerRadius = 0.55 * EwHubOuterRadius;
EwAngularPitch = 360 / EwToothCount;
EwInterSpokeAngle = 360 / EwSpokesCount;
```

```
/*
   The Deadbeat Fork parameters.
 */
ForkToothSpanCount = 7.5;
ForkEntryArmAngle = 35;
ForkExitArmAngle = 32;
ForkLockAngle = 1;
ForkDropAngle = 1;
ForkLiftAngle = 2;

ForkPalletAngularWidth = EwAngularPitch / 2 - ForkDropAngle;

ForkWheelPalletAngle = ( EwAngularPitch * ForkToothSpanCount ) / 2;
```

```
ForkWheelDistance = EwTipRadius / cos( ForkWheelPalletAngle );

ForkOuterRadius = sqrt( pow( EwTipRadius, 2 ) +
      pow( ForkWheelDistance, 2 ) -
      2 * EwTipRadius * ForkWheelDistance *
      cos( ForkWheelPalletAngle + ForkPalletAngularWidth / 2 ) );

ForkInnerRadius = sqrt( pow( EwTipRadius, 2 ) +
      pow( ForkWheelDistance, 2 ) -
      2 * EwTipRadius * ForkWheelDistance *
      cos( ForkWheelPalletAngle - ForkPalletAngularWidth / 2 ) );

ForkPalletLinearWidth = ForkOuterRadius - ForkInnerRadius;

LnFW = asin( ( EwTipRadius / ForkOuterRadius ) *
    sin( ForkWheelPalletAngle + ForkPalletAngularWidth / 2 ) );

CnFW = LnFW - ForkLockAngle;

ImpFcX = ForkOuterRadius * cos( CnFW );
ImpFcY = ForkOuterRadius * sin( CnFW );

CnDn = sqrt( pow( ForkOuterRadius, 2 ) +
      pow( ForkInnerRadius, 2 ) -
      2 * ForkOuterRadius * ForkInnerRadius *
      cos( ForkLiftAngle ) );

DnCnF = asin( (ForkInnerRadius / CnDn ) * sin( ForkLiftAngle ) );

ForkArmWidth = ForkPalletLinearWidth;
ForkHubOuterRadius = 5 * BSU;
ForkHubInnerRadius = 0.5 * ForkHubOuterRadius;
FokrArmLength = ForkOuterRadius - ForkHubInnerRadius;

ew( EwTipRadius,
      EwRootRadius,
      EwRimWidth,
      EwToothCount,
      EwToothTrailingAngle,
      EwToothIncludedAngle,
      EwSpokesCount,
      EwHubOuterRadius,
      EwHubInnerRadius );

/*
   The fork's axis of symmetry is along the OX axis.
 */
translate( [ ForkWheelDistance, 0, 0 ] )
dbfork( ForkHubOuterRadius, ForkHubInnerRadius );
```