Tech document

1. GMT

   根據不同 Mode 的軌道，做對應的矩陣乘法

```
Pnt3f TrainView::GMT(Pnt3f p1, Pnt3f p2, Pnt3f p3, Pnt3f p4, float mode, float t) {
    glm::mat4x4 G = {
        {p1.x,p2.x,p3.x,p4.x},
        {p1.y,p2.y,p3.y,p4.y},
        {p1.z,p2.z,p3.z,p4.z},
        {1,1,1,1}
    };
    G = glm::transpose(G);
    glm::mat4x4 M;
    if (mode == 1) {
        M = { 0, 0, 0, 0,
              0, 0, -1, 1,
              0, 0, 1, 0,
              0, 0, 0, 0 };
    }
    else if (mode == 2) {
        M = {
                {-1.0f,2.0f,-1.0f,0.0f},
                {2.0f/tense-1.0f,1.0f-3.0f/tense,0.0f,1.0f/tense},
                {1.0f -2.0f / tense,3.0f/tense-2.0f,1.0f,0.0f},
                {1.0f,-1.0f,0.0f,0.0f}
        };
        M *= tense;
    }
    else if (mode == 3) {
        M = {
            {-1.0f,3.0f,-3.0f,1.0f},
            {3.0f,-6.0f,0.0f,4.0f},
            {-3.0f,3.0f,3.0f,1.0f},
            {1.0f,0.0f,0.0f,0.0f}
        };
        M /= 6.0f;
    }
    M = glm::transpose(M);
    glm::vec4 T = { pow(t,3),pow(t,2),pow(t,1),pow(t,0) };
    glm::vec4 result = G * M * T;
    return Pnt3f(result[0], result[1], result[2]);
}
```

2. 火車移動-參數化

   根據 q1-q0 的距離累加，判斷火車行走的距離，最後儲存火車應該在哪個位置

```
for (size_t i = 0; i < m_pTrack->points.size(); i++) {
    float t = 0;
    ControlPoint p1 = m_pTrack->points[(i - 1 + m_pTrack->points.size()) % m_pTrack->points.size()];
    ControlPoint p2 = m_pTrack->points[(i + m_pTrack->points.size()) % m_pTrack->points.size()];
    ControlPoint p3 = m_pTrack->points[(i + 1 + m_pTrack->points.size()) % m_pTrack->points.size()];
    ControlPoint p4 = m_pTrack->points[(i + 2 + m_pTrack->points.size()) % m_pTrack->points.size()];
    for (size_t j = 0; j < DIVIDE_LINE; j++) {
        Pnt3f qt0 = GMT(p1.pos, p2.pos, p3.pos, p4.pos, TrainV->tw->splineBrowser->value(), t);
        Pnt3f orient_t= GMT(p1.orient, p2.orient, p3.orient, p4.orient,TrainV->tw->splineBrowser->value(), t);
        Pnt3f qt1 = GMT(p1.pos, p2.pos, p3.pos, p4.pos, TrainV->tw->splineBrowser->value(), t += percent);
        Pnt3f forward = qt1 - qt0;
        Pnt3f cross_t = forward * orient_t;
        cross_t.normalize();
        orient_t = cross_t * forward;
        orient_t.normalize();
        cross_t = cross_t * Sleeper_Width;
```

```
Path_Total += sqrt(forward.x * forward.x + forward.y * forward.y + forward.z * forward.z);
if (!check && Path_Total > TrainV->m_pTrack->trainU) {
    if (qt1.y - qt0.y > 0) {
        physical = qt0.y - qt1.y;
    }
    else if (qt1.y - qt0.y < 0) {
        physical = qt0.y - qt1.y;
        physical *= 0.7;
    }
    else {
        physical = 0;
    }
    t_t = t;
    t_i = i;
    check = true;
}
```

```
if (trainView->tw->arcLength->value()) {
    m_Track.trainU += 1.0f * speed->value()+ trainView->physical*10;
    if (m_Track.trainU > trainView->Path_Total) {
        m_Track.trainU -= trainView->Path_Total;
    }
}
```

3. 火車移動-非參數化

   讓 t_time 累加移動

```
for (size_t i = 0; i < m_pTrack->points.size(); i++) {
    float t = 0;
    ControlPoint p1 = m_pTrack->points[(i - 1 + m_pTrack->points.size()) % m_pTrack->points.size()];
    ControlPoint p2 = m_pTrack->points[(i + m_pTrack->points.size()) % m_pTrack->points.size()];
    ControlPoint p3 = m_pTrack->points[(i + 1 + m_pTrack->points.size()) % m_pTrack->points.size()];
    ControlPoint p4 = m_pTrack->points[(i + 2 + m_pTrack->points.size()) % m_pTrack->points.size()];
    for (size_t j = 0; j < DIVIDE_LINE; j++) {
        Pnt3f qt0 = GMT(p1.pos, p2.pos, p3.pos, p4.pos, TrainV->tw->splineBrowser->value(), t);
        Pnt3f orient_t= GMT(p1.orient, p2.orient, p3.orient, p4.orient,TrainV->tw->splineBrowser->value(), t);
        Pnt3f qt1 = GMT(p1.pos, p2.pos, p3.pos, p4.pos, TrainV->tw->splineBrowser->value(), t += percent);
        Pnt3f forward = qt1 - qt0;
        Pnt3f cross_t = forward * orient_t;
        cross_t.normalize();
        orient_t = cross_t * forward;
        orient_t.normalize();
        cross_t = cross_t * Sleeper_Width;
```

```
else {
    if (trainView->tw->arcLength->value() != trainView->s) {
        trainView->t_time = trainView->t_t + trainView->t_i;
    }
    trainView->t_time += (dir * speed->value()/100);
    m_Track.trainU += 1.0f * speed->value();
}
if (trainView->t_time >= trainView->m_pTrack->points.size()) {
    trainView->t_time -= trainView->m_pTrack->points.size();
}
```

4. 鐵軌

   根據 q1-q0 的距離累加，超過一定的值就畫，確保間距離相同

```
for (size_t i = 0; i < m_pTrack->points.size(); i++) {
    float t = 0;
    ControlPoint p1 = m_pTrack->points[(i - 1 + m_pTrack->points.size()) % m_pTrack->points.size()];
    ControlPoint p2 = m_pTrack->points[(i + m_pTrack->points.size()) % m_pTrack->points.size()];
    ControlPoint p3 = m_pTrack->points[(i + 1 + m_pTrack->points.size()) % m_pTrack->points.size()];
    ControlPoint p4 = m_pTrack->points[(i + 2 + m_pTrack->points.size()) % m_pTrack->points.size()];
    for (size_t j = 0; j < DIVIDE_LINE; j++) {
        Pnt3f qt0 = GMT(p1.pos, p2.pos, p3.pos, p4.pos, TrainV->tw->splineBrowser->value(), t);
        Pnt3f orient_t= GMT(p1.orient, p2.orient, p3.orient, p4.orient,TrainV->tw->splineBrowser->value(), t);
        Pnt3f qt1 = GMT(p1.pos, p2.pos, p3.pos, p4.pos, TrainV->tw->splineBrowser->value(), t += percent);
        Pnt3f forward = qt1 - qt0;
        Pnt3f cross_t = forward * orient_t;
        cross_t.normalize();
        orient_t = cross_t * forward;
        orient_t.normalize();
        cross_t = cross_t * Sleeper_Width;
```

```
Sleep_Total += sqrt(forward.x * forward.x + forward.y * forward.y + forward.z * forward.z);
if (!Draw_Sleeper && Sleep_Total >= Sleeper_Length) {
    count++;
    forward.normalize();
    DrawSleeper(qt0, qt0 - forward * Sleeper_Length, cross_t, orient_t, doingShadows);
    if (count % 3 == 0) {
        DrawPillar(qt0, qt0 - forward * Sleeper_Length, cross_t, orient_t, doingShadows);
        count = 0;
    }
    Sleep_Total -= Sleeper_Length;
    Draw_Sleeper = !Draw_Sleeper;
}
else if (Draw_Sleeper && Sleep_Total >= Sleeper_Interval) {
    Sleep_Total -= Sleeper_Interval;
    Draw_Sleeper = !Draw_Sleeper;
}
```

5. Add/Delete Car

往後推算 Add 的車需要畫在哪個軌道之間哪個 t 值

參數化:

```
while (!check) {
    for (t; t >= 0; t =t- percent) {
        qt0 = GMT(p1.pos, p2.pos, p3.pos, p4.pos, TrainV->tw->splineBrowser->value(), t);
        qt1 = GMT(p1.pos, p2.pos, p3.pos, p4.pos, TrainV->tw->splineBrowser->value(), t + percent);
        forward = qt1 - qt0;
        total += sqrt(forward.x * forward.x + forward.y * forward.y + forward.z * forward.z);
        if (total >= 13) {
            check = true;
            break;
        }
    }
    if (check) {
        break;
    }
    t = 1.0f;
    i-=1;
    if (i < 0) {
        i = m_pTrack->points.size() - 1;
    }
    p1 = m_pTrack->points[(i - 1 + m_pTrack->points.size()) % m_pTrack->points.size()];
    p2 = m_pTrack->points[(i + m_pTrack->points.size()) % m_pTrack->points.size()];
    p3 = m_pTrack->points[(i + 1 + m_pTrack->points.size()) % m_pTrack->points.size()];
    p4 = m_pTrack->points[(i + 2 + m_pTrack->points.size()) % m_pTrack->points.size()];
}
```

非參數化:

直接-t 值往後推

```
else {
    float tt = t_time - percent * 200 * (j - 1);
    int ii = floor(tt);
    tt -= ii;
    ControlPoint p1 = m_pTrack->points[(ii - 1 + m_pTrack->points.size()) % m_pTrack->points.size()];
    ControlPoint p2 = m_pTrack->points[(ii + m_pTrack->points.size()) % m_pTrack->points.size()];
    ControlPoint p3 = m_pTrack->points[(ii + 1 + m_pTrack->points.size()) % m_pTrack->points.size()];
    ControlPoint p4 = m_pTrack->points[(ii + 2 + m_pTrack->points.size()) % m_pTrack->points.size()];

    Pnt3f qt0 = GMT(p1.pos, p2.pos, p3.pos, p4.pos, TrainV->tw->splineBrowser->value(), tt);
    Pnt3f orient_t = GMT(p1.orient, p2.orient, p3.orient, p4.orient, TrainV->tw->splineBrowser->value(), tt);
    Pnt3f qt1 = GMT(p1.pos, p2.pos, p3.pos, p4.pos, TrainV->tw->splineBrowser->value(), tt += percent);

    Pnt3f cross_t = (qt1 - qt0) * orient_t;
    cross_t.normalize();
    orient_t = cross_t * (qt1 - qt0);
    orient_t.normalize();
    cross_t = cross_t * Train_Width;
    Pnt3f up = orient_t * Train_Height;
    Pnt3f forward = (qt1 - qt0);
    forward.normalize();
    forward = forward * Train_Forward;
    if (j == 1) {
        DrawTrainHead(qt0, cross_t, up, forward, doingShadows);
    }
    else {
        DrawTrain(qt0, cross_t, up, forward, doingShadows);
    }
}
```