## Overview

- Representing Polygon Meshes
  - Explicit
  - Pointers to a vertex list
  - Pointers to an edge list
- Parametric Cubic Curves
  - Hermite Curves
  - Bezier Curves
  - B-Spline Curves
- NURBS

## Representing Polygon Meshes

**Polygon mesh -** a collection of edges, vertices and polygons connected such that each edge is shared by at most two polygons.

Polygon meshes can be represented many different ways and are evaluated according to space and time.

## Explicit Representation

Each polygon is represented by a list of vertex coordinates.

$$P = ((x_1, y_1, z_1), (x_2, y_2, z_2), \cdots, (x_n, y_n, z_n))$$

**Takes Space** - for more than one polygon space is wasted, because vertices are duplicated.

**Takes Time** - since there is no explicit representation of edges and vertices, an interactive move of a vertex involves finding all polygons that share the vertex.

**Display** - the shared edges are drawn twice which can cause problems on pen plotters. Extra pixels can be lit when edges are draw in opposite direction.

## Pointers to a Vertex List

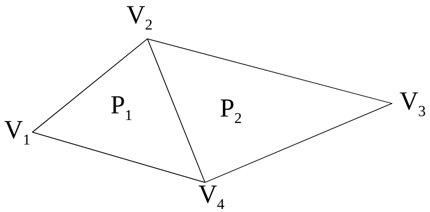
Each vertex is stored once in a vertex list

$$V = (V_1, V_2, V_3, V_4) = ((x_1, y_1, z_1), \cdots, (x_4, y_4, z_4))$$

$P_1 = (1,2,4)$

$P_2 = (4,2,3)$

P is a list of indices into a vertex list.



## Pointers to a Vertex List

**Advantages -**

Space saved because each vertex is stored once.

Coordinates of a vertex can be changed easily.

**Disadvantage -**

Difficult to find polygons that share edges.

Draws polygon edges twice.

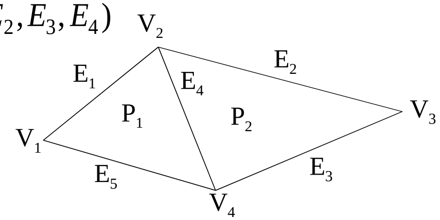
## Pointers to an Edge List

A polygon is represented by a pointer to the edge list.

$$V = (V_1, V_2, V_3, V_4) = ((x_1, y_1, z_1), \cdots, (x_4, y_4, z_4))$$

$$E_1 = (V_1, V_2, P_1, \lambda)$$
$$E_2 = (V_2, V_3, P_2, \lambda)$$
$$E_3 = (V_3, V_4, P_2, \lambda)$$
$$E_4 = (V_4, V_2, P_1, P_2)$$
$$E_5 = (V_4, V_1, P_1, \lambda)$$

$$P_1 = (E_1, E_4, E_5)$$
$$P_2 = (E_2, E_3, E_4)$$




## Pointers to an Edge List

**Advantages -**

Displays edges rather than polygons.

Eliminates redundant clipping, transformation and scan conversion.

Filled polygons are more easily clipped.

In all three cases, the determining of which edges are incident to a vertex is not easy. All edges must be inspected.

## Plane Equation

The plane equation can be found by using the coordinates of three vertices.

$$Ax + By + Cz + D = 0$$

Where *A*, *B*, and *C* define the normal to the plane and (*x*, *y*, *z*) is any point on the plane.

The planes normal can be computed as the cross product between three points on the plane

$$P_1P_2 \times P_1P_3$$

A nonzero cross product defines a plane and D can be found by substitution.

## Parametric Cubic Curves

- Cubic are a good degree because:
  - It is high enough to allow some flexibility in the curve design.
  - It is not so high that wiggles creep into the curve.
  - It is the lowest degree that can specify a non-planar space curve.
  - A compromise between flexibility and speed of computation.

## Parametric Cubic Curves

Parametric Representation:

$$x = x(t) \qquad y = y(t) \qquad z = z(t)$$

The cubic polynomials that define a curve segment.

$$x(t) = a_x t^3 + b_x t^2 + c_x t + d_x,$$
$$y(t) = a_y t^3 + b_y t^2 + c_y t + d_y,$$
$$z(t) = a_z t^3 + b_z t^2 + c_z t + d_z, 0 \le t \le 1$$

## Parametric Cubic Curves

$$Q(t) = \begin{bmatrix} x(t) & y(t) & z(t) \end{bmatrix} = T \cdot C, where$$

$$T = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \quad \text{and} \quad C = \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix}$$

The parametric tangent vector of the curve

$$\frac{d}{dt}Q(t) = \frac{d}{dt}T \cdot C = \begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix} \cdot C$$

Needed for continuity

## Parametric Cubic Curves

The coefficient matrix **C** can be written as **C = G·M**, where M is a 4x4 basis matrix. G is a four element matrix of geometric constraints (geometry matrix).

$$Q(t) = G \cdot M \cdot T$$

$$Q(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} = \begin{bmatrix} G_1 & G_2 & G_3 & G_4 \end{bmatrix} \begin{bmatrix} m_{11} & m_{21} & m_{31} & m_{41} \\ m_{12} & m_{22} & m_{32} & m_{42} \\ m_{13} & m_{23} & m_{33} & m_{43} \\ m_{14} & m_{24} & m_{34} & m_{44} \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}$$

## Parametric Cubic Curves

The **blending function B** are given by B = M · T.

$$Q(t) = G \cdot B$$

A curve segment $Q(t)$ is defined by constraints on **endpoints**, **tangent vectors** and **continuity** between curve segments.

## Parametric Cubic Curves

Three major curve types:

**Hermite -**
defined by two endpoints and two endpoint tangent vectors.

**Bézier -**
defined by two endpoints and two other points that control the endpoint tangent vector.

**B-Spline -**
defined by four control points and has $C^1$ and $C^2$ continuity at the join points. Does not generally interpolate the control points.

## Cubic Hermite Curves

The **Hermite geometry vector $\mathbf{G_H}$** represents the four constraints of the Hermite curve.
The x component is: $G_{H_x} = [P_{1_x} \quad P_{4_x} \quad R_{1_x} \quad R_{4_x}]$

$$x(t) = G_{H_x} \cdot M_H \cdot T = G_{H_x} \cdot M_H \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix}^T$$

Need to find the **Hermite basis matrix $\mathbf{M_H}$**:
The constraints on x(0) and x(1) (the end points) can be found by substitution:

$$x(0) = P_{1_x} = G_{H_x} \cdot M_H \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T$$

$$x(1) = P_{4_x} = G_{H_x} \cdot M_H \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}^T$$

## Cubic Hermite Curves

The tangent vector constraint can be found by differentiation:

$$x'(0) = R_{1_x} = G_{H_x} \cdot M_H \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}^T$$

$$x'(1) = R_{4_x} = G_{H_x} \cdot M_H \begin{bmatrix} 3 & 2 & 1 & 0 \end{bmatrix}^T$$

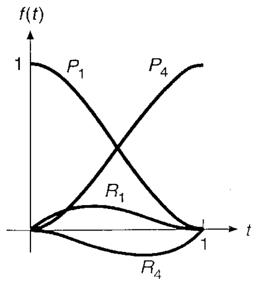
The **Hermite basis matrix $\mathbf{M_H}$** is the inverse of the 4x4 matrix from the constraints.

$$M_H = \begin{bmatrix} 0 & 1 & 0 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}^{-1} = \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix}$$

## Cubic Hermite Curves

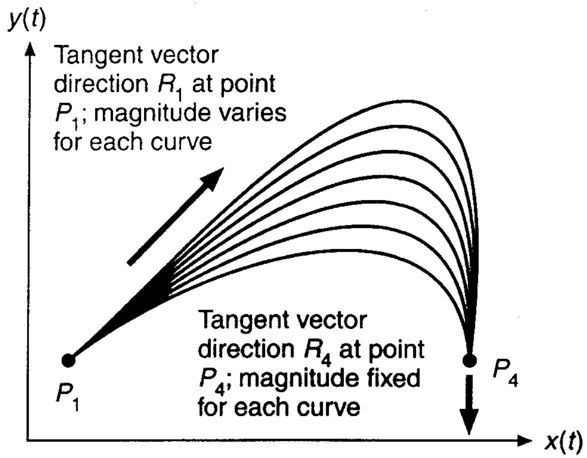
$$x(t) = G_{H_x} \cdot M_H \cdot T =$$

$$(2t^3 - 3t^2 + 1)P_1 + (-2t^3 + 3t^2)P_4 + (t^3 - 2t^2 + t)R_1 + (t^3 - t^2)R_4$$




The Hermite blending functions, labeled by the elements of the geometry vector that they weight.

## Cubic Hermite Curves




## Cubic Bézier Curves

The Bézier **geometry matrix $\mathbf{G_B}$** consists of four control points.

$$G_B = [P_1 \quad P_2 \quad P_3 \quad P_4]$$

The Bézier **basis matrix $\mathbf{M_B}$** is found by substitution:

$$Q(t) = (G_B \cdot M_{HB}) \cdot M_H \cdot T = G_B \cdot (M_{HB} \cdot M_H) \cdot T = G_B \cdot M_B \cdot T$$

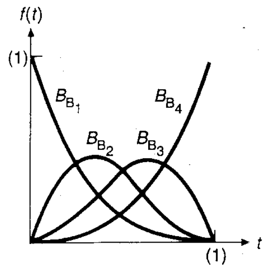
$$M_B = \begin{bmatrix} 1 & 0 & -3 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & -3 \\ 0 & 1 & 0 & 3 \end{bmatrix} \cdot \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

## Cubic Bézier Curves

The Bézier **blending functions $\mathbf{B_B}$** are called the Bernstein polynomials

$$Q(t) = G_B \cdot M_B \cdot T = G_B \cdot B_B =$$

$$(1-t)^3 P_1 + 3t(1-t)^2 P_2 + 3t^2(1-t)P_3 + t^3 P_4$$




The Bernstein polynomials, which are the weighting functions for Bézier curves. At $t = 0$, only $B_{B_1}$ is nonzero, so the curve interpolates $P_1$; similarly, at $t = 1$, only $B_{B_4}$ is nonzero, and the curve interpolates $P_4$.

# Properties of the Bézier Curve

**The blending functions -**

are non-negative and they all sum to unity

**Convex hull property -**

for t∈[0,1], each curve segment is completely within the convex hull of the four control points.

**Symmetry**

**Endpoint interpolation**

A Bézier curve can have $C^0$ and $C^1$ continuity at the join points (the three points must be distinct and collinear)

# Bézier Curve Algorithms

Bézier curves satisfy the following recursion: de Casteljau algorithm

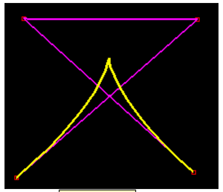
$$B_i^r(t) = (1-t)B_i^{r-1}(t) + tB_{i+1}^{r-1}(t)$$

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

Bernstein polynomials

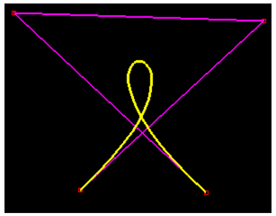
$$\binom{n}{i} = \begin{cases} \dfrac{n!}{i!(n-i)!} & if \quad 0 \le i \le n \\ 0 & else \end{cases}$$

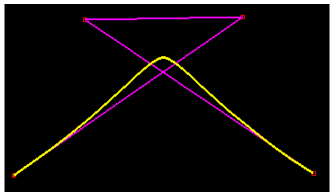
# Cubic Bézier Curves



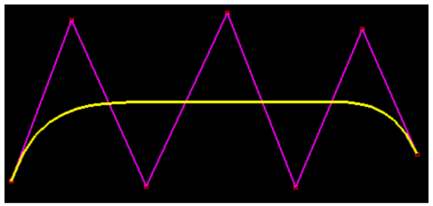
cusp

Degree 3 curves

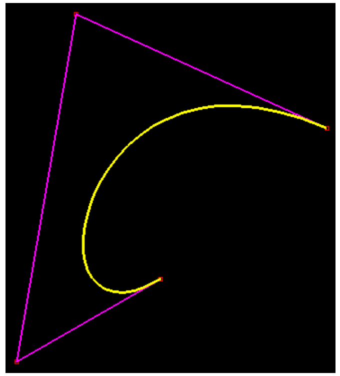




# Bézier Curves



Degree 6 curve



## Higher Degree Curves

- A single cubic Bezier or Hermite curve can only capture a small class of curves.
- One solution is to raise the degree.
  - Allows more control, at the expense of more control points and higher degree polynomials.
  - Control is not *local*, one control point influences entire curve
- Alternate, most common solution is to join pieces of cubic curves together into ***piecewise cubic curves***
  - Total curve can be broken into pieces, each of which is cubic.
  - ***Local control***: Each control point only influences a limited part of the curve.
  - Interaction and design is much easier.

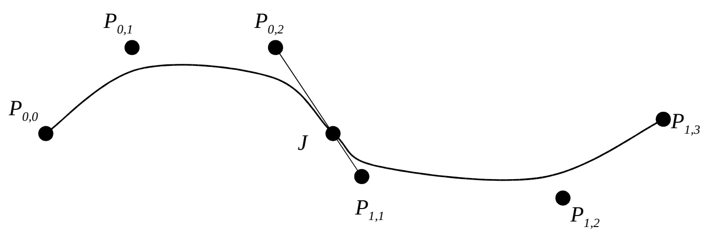
## Geometric Continuity

$G^0$ - when two curve segments join (same coordinate position).

$G^1$ - when two curve segments have equal tangent vectors at the join point (1st derivative). E.g., $TV_1 = kTV_2$ (same direction).

$G^2$ - when both first and second parametric derivative of the curve sections are proportional at their boundary.

## Bézier Geometric Continuity




## Parametric Continuity

$C^0$ - when two curve segments join (same coordinate position).

$C^1$ - when the tangent vectors at the curves join point are equal (direction and magnitude) (1st derivative).

$C^n$ - when direction and magnitude of $\frac{d^n}{dt^n}[Q(t)]$ through the $n^{th}$ derivative are equal at the join point.

In general, $C^1$ continuity implies $G^1$, but the converse is generally not true.

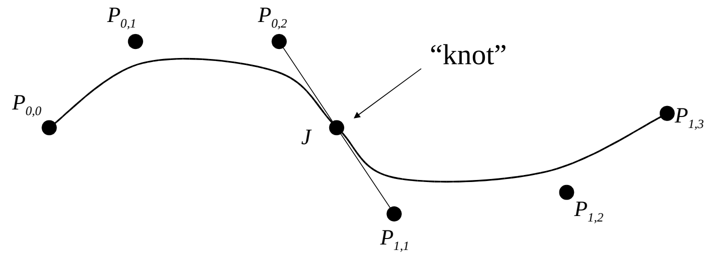
$$C^1 \Rightarrow G^1$$

$C^n$ continuity is more restrictive than $G^n$ continuity.

# Achieving Continuity

- For Hermite curves, the user specifies the derivatives, so $C^1$ is achieved simply by sharing points and derivatives across the “knot”.
- For Bezier curves:
  - They interpolate their endpoints, so $C^0$ is achieved by sharing control points
  - The parametric derivative is a constant multiple of the vector joining the first/last 2 control points
  - So $C^1$ is achieved by setting $P_{0,3}=P_{1,0}=J$, and making $P_{0,2}$ and $J$ and $P_{1,1}$ collinear, with $J-P_{0,2}=P_{1,1}-J$

# Bézier Parametric Continuity

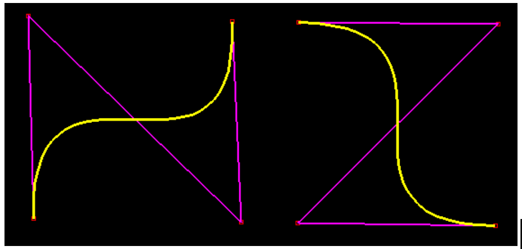



Disclaimer: PowerPoint curves are not Bezier curves, they are interpolating piecewise quadratic curves! This diagram is an approximation.

# Invariance

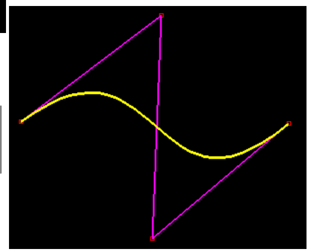
- ***Translational invariance*** means that translating the control points and then evaluating the curve is the same as evaluating and then translating the curve.
- ***Rotational invariance*** means that rotating the control points and then evaluating the curve is the same as evaluating and then rotating the curve.
- These properties are essential for parametric curves used in graphics.
- Bezier curves, Hermite curves and B-splines are translational and rotational invariant.
- Some curves, *rational splines* (eg. NURBS), are also ***perspective invariant***
  - Can do perspective transform of control points and *then* evaluate the curve.

# Bézier Curves



**Invariant under rotations**

**Symmetry**



# Cubic B-Splines

- Uniform Non-rational B-Splines
  - Knot are spaced at equal interval of the parameter t.
- Non-uniform Non-rational B-Splines
  - The parameter interval between the knot values is not necessarily uniform.
- Non-uniform Rational B-Splines (NURBS)

# Cubic B-Splines

The B-spline geometry matrix $G_{B_{Si}}$ for segment $Q_i$

$$G_{B_{Si}} = \begin{bmatrix} P_{i-3} & P_{i-2} & P_{i-1} & P_i \end{bmatrix},\ 3 \le i \le m$$

$T_i$ is the column vector

$$\begin{bmatrix} (t-t_i)^3 & (t-t_i)^2 & (t-t_i) & 1 \end{bmatrix}^T$$

The B-spline formulation for a curve segment is

$$Q_i(t) = G_{B_{Si}} \cdot M_{B_{Si}} \cdot T_i,\ \ t_i \le t \le t_{i+1}$$

The B-spline basis matrix, $M_{B_{Si}}$ , is $M_{B_S} = \frac{1}{6}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}$

The blending function is defined $B_{B_S} = M_{B_S} \cdot T$

# Properties of Cubic B-Splines

**The blending functions -**
are non-negative and they all sum to unity.

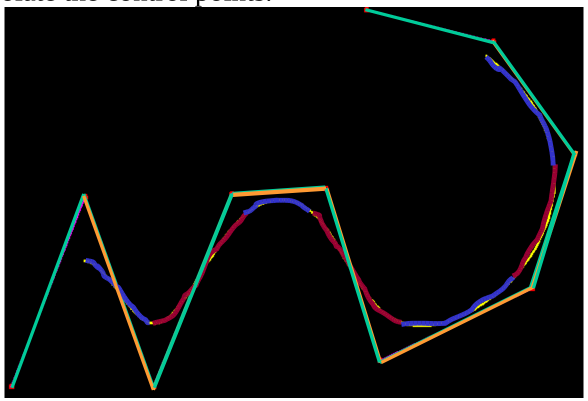
**Convex hull property -**
for $t \in [0,1]$, each point on the curve lies completely within the convex hull of the control polygon.

**Local control -**
moving a control point affects only the four curve segments the control point controls. This makes B-Splines more flexible than Bézier curves.

# Uniform Nonrational B-spline

B-spline curves are $C^0$, $C^1$ and $C^2$ continuous cubic polynomials that do not interpolate the control points.

**$m = 9$, $m \ge 3$**
**$m+1$**
**control points**
**($P_1, P_2, \ldots, P_{m+1}$)**
**$m-2$**
**curve segments**
**($Q_3, Q_4, \ldots, Q_m$)**
**$m-1$ knots**
**$Q_i$ is defined**
**$t_i \le t \le t_{i+1}$, for**
**$3 \le i \le m$**



# Nonuniform Nonrational B-Spline

- The parameter interval between the knot values is not necessarily uniform.
- Blending functions vary for each interval, but they are non-negative and sum to unity.
- Each curve segment is in the convex hull.
- The de Boor algorithm is used to display B-spline curves.

Let $t \in [t_I, t_{I+1}]$

$$d_i^k(t) = \frac{t_{i+n-k} - t}{t_{i+n-k} - t_{i-1}} d_{i-1}^{k-1}(t) + \frac{t - t_{i-1}}{t_{i+n-k} - t_{i-1}} d_i^{k-1}(t)$$

$$for \quad k = 1,\ldots,n-r, \quad and \quad i = I-n+k+1,\ldots,I+1$$

# Nonuniform Nonrational B-Spline

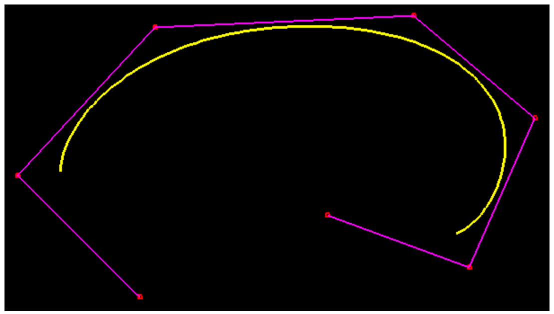
**Advantage over uniform B-spline:**

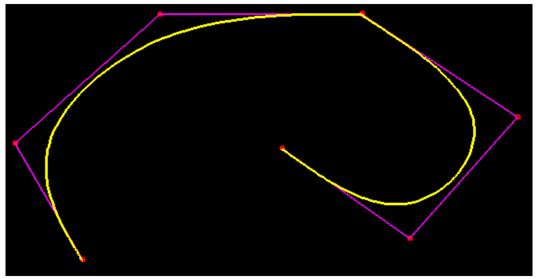
- continuity at the join points can be reduced from $C^2$ to $C^1$ to $C^0$ to none, by using **multiple knots**.

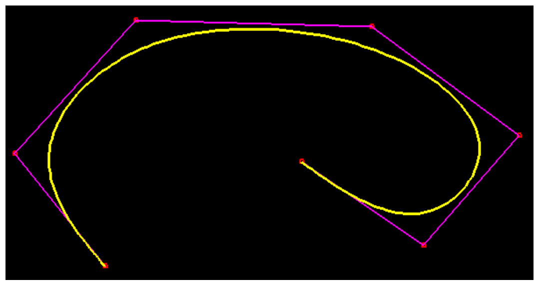
- $C^0$ continuity means that the curve interpolates a control point (do not get a straight line on each side of the interpolated control point).

- start point and endpoint can easily be interpolated without introducing linear segments.

- a knot (and a control point) can be inserted so the curve can easily be reshaped.







| | Hermite | Bezier | Uniform B-Spline | Nonuniform B-Spline |
|---|---|---|---|---|
| Convex Hull defined by points | N/A | YES | YES | YES |
| Interpolates some control points | YES | YES | NO | NO |
| Interpolates all control points | YES | NO | NO | NO |
| Ease of Subdivision | Good | Best | Average | High |
| Continuities inherent in representation | $C^0$ $G^0$ | $C^0$ $G^0$ | $C^2$ $G^2$ | $C^2$ $G^2$ |
| Continuities achieved easily | $C^1$ $G^1$ | $C^1$ $G^1$ | $C^2$ $G^2$ | $C^2$ $G^2$ |