What's New in Maple 2017

# Graph Theory

There have been several updates for the GraphTheory package in Maple 2017, including an update to the DrawGraph command to use a grayscale color scheme for graphs and the ability to control the graph drawing styles as well as several new commands.

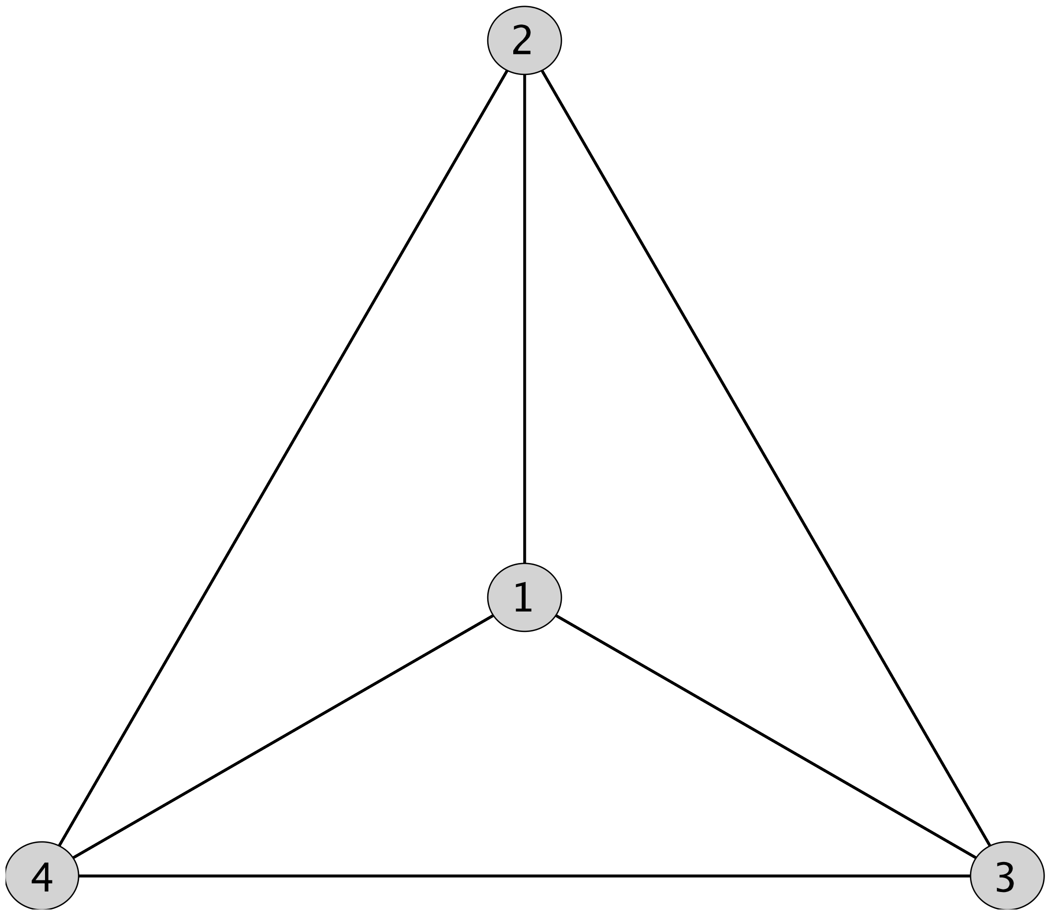
## Graph Drawing Options

By default, graphs in Maple 2017 are drawn using a grayscale color scheme.

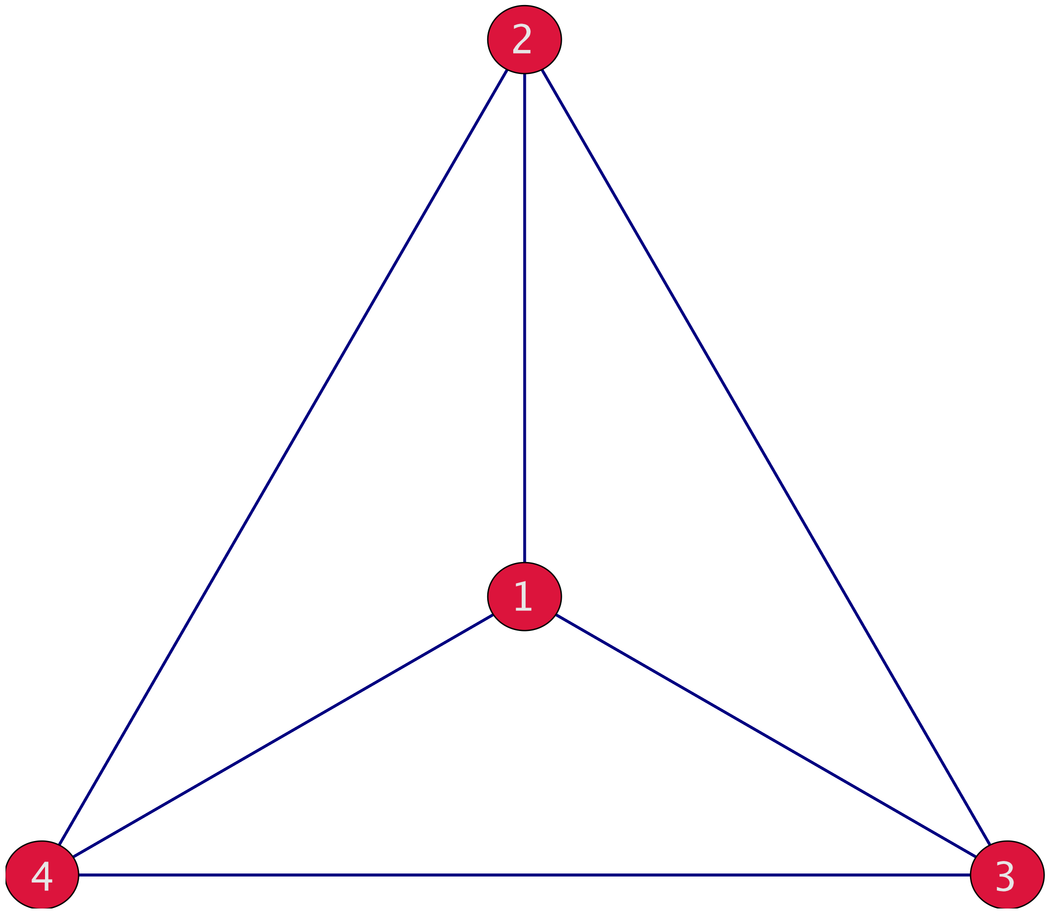
You can now control many aspects of how a graph is drawn. The colors, line weights, and font choices can be specified with the stylesheet option.

```
with(GraphTheory) :
```

$DrawGraph(SpecialGraphs{:-}TetrahedronGraph(\,))$;



*DrawGraph*(*SpecialGraphs*:-*TetrahedronGraph*( ), *stylesheet* = [*edgecolor* = "NavyBlue",
    *vertexcolor* = "Crimson"]);



You can use the graph drawing settings from previous versions of Maple by specifying stylesheet="legacy". Animations and 3-D graph renderings currently do not support the stylesheet option.

## ▼ Automorphism Groups

### ▼ AutomorphismGroup

The new GraphTheory[AutomorphismGroup] command computes and returns the group of automorphisms of a given graph, represented as a permutation group.

```
with(GraphTheory) : with(GroupTheory) :
PG := SpecialGraphs:-PetersenGraph( )
          PG := Graph 1: an undirected unweighted graph with 10 vertices and 15 edge(s)
AG := AutomorphismGroup(PG)
AG := ⟨(1, 2)(3, 5)(6, 9)(7, 8), (2, 5)(3, 4)(7, 10)(8, 9), (3, 9)(4, 8)(7, 10), (4, 7)(5,
    6)(8, 10)⟩
```

After the automorphism group is computed, you can use GroupTheory commands to analyze the computed group. Here we use GroupTheory[IdentifySmallGroup] and GroupTheory[AreIsomorphic] to confirm the identity of the generated group and demonstrate it is, in fact, isomorphic to symmetric group on 5 elements:

```
IdentifySmallGroup(AG)
                                  120, 34
AreIsomorphic(AG, SymmetricGroup(5))
                                   true
```

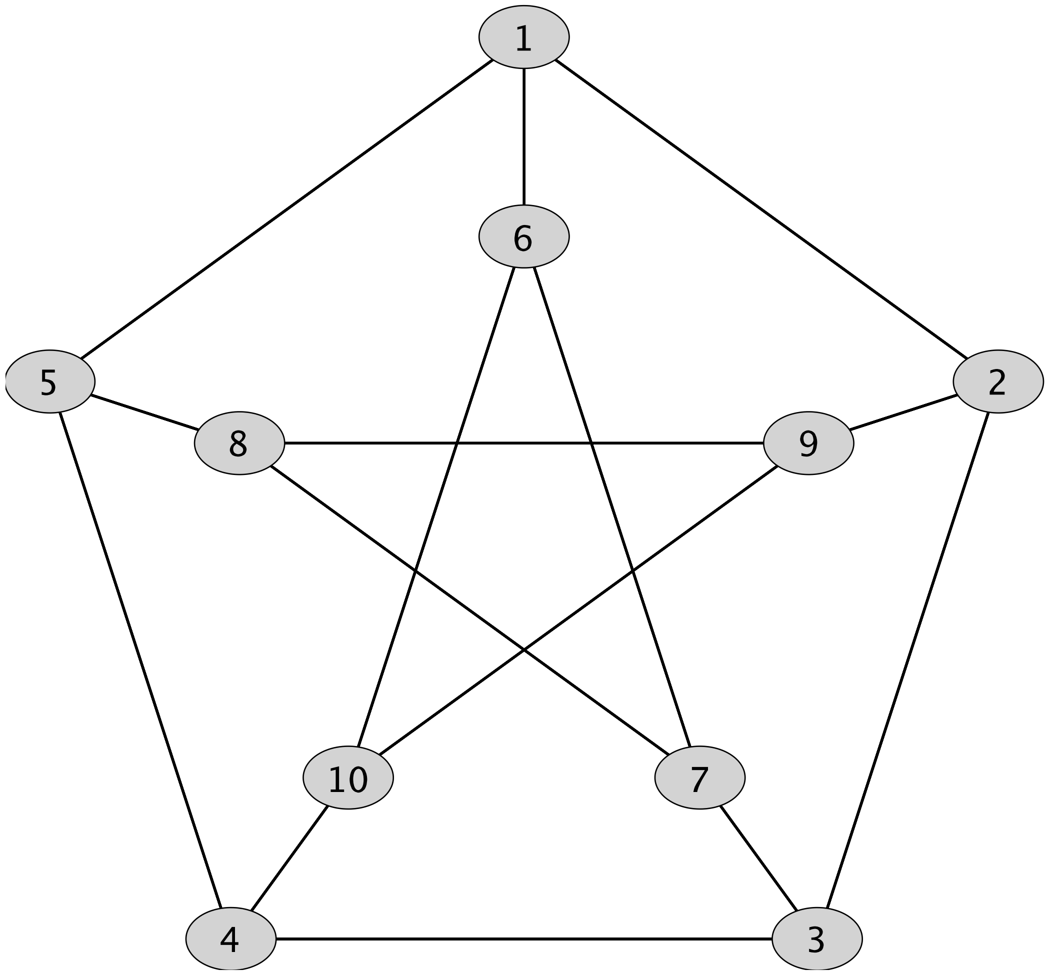
## DrawAutomorphism

The new GraphTheory[DrawAutomorphism] command enables visualizing the action of an automorphism of a graph as an animation.
When given a graph and a permutation corresponding to an automorphism, DrawAutomorphism produces an animation showing the action of each of the generators of the graph's automorphism group.

In the example below, we extract the generators of the computed automorphism group and create a permutation corresponding to a particular graph automorphism by multiplying all four generators. We then instruct **DrawAutomorphismGroup** to show this particular automorphism.

```
g := Generators(AG)
g := [(1, 2)(3, 5)(6, 9)(7, 8), (2, 5)(3, 4)(7, 10)(8, 9), (3, 9)(4, 8)(7, 10), (4, 7)(5, 6)(8,
    10)]
p := g[1] · g[2] · g[3] · g[4]
                      p := (1, 6, 7, 3, 2)(4, 9, 5, 10, 8)
```

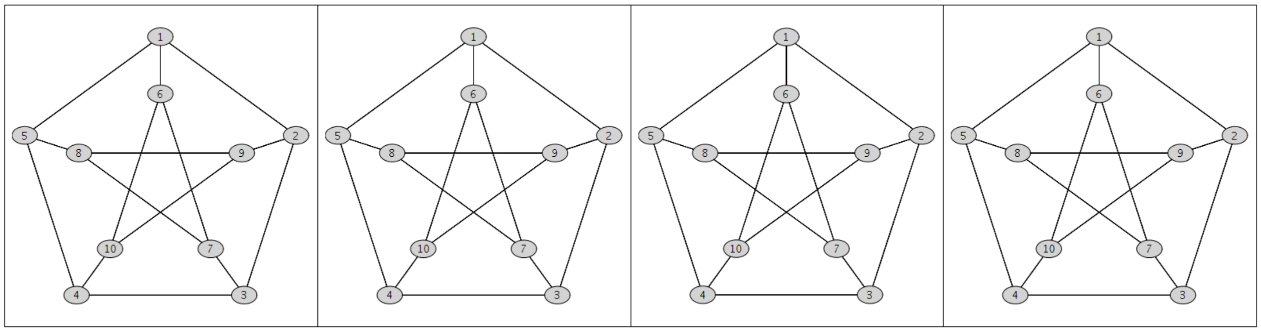
*DrawAutomorphism*$(PG, p)$



$[6, 1, 2, 9, 10, 7, 3, 4, 5, 8]$

If one provides only the graph, DrawAutomorphism shows an animation of the automorphisms corresponding to each of the generators of the automorphism group, as returned by **AutomorphismGroup**.

*DrawAutomorphism*$(PG)$



## New Commands: CanonicalGraph, Eccentricity, Radius

The new commands GraphTheory[CanonicalGraph], GraphTheory[Eccentricity], and GraphTheory[Radius] enable the construction of new graph normal forms and the computation of quantities from graphs.

The **CanonicalGraph** command constructs a version of the input graph in which the vertices have been reordered such that the resulting graph is in a canonical form. The output is canonical in the sense that any two graphs $G$ and $H$ are isomorphic if and only if $AdjacencyMatrix(CanonicalGraph(G)) = AdjacencyMatrix(CanonicalGraph(H))$. In this example, the Foster cage graph and the Meringer graph serve as examples of graphs which are both cage graphs with the same number of vertices and edges, but are nevertheless not isomorphic.

```
CanonicalGraph(SpecialGraphs:-FosterCageGraph( ))
                    Graph 2: an undirected unweighted graph with 30 vertices and 75 edge(s)
CanonicalGraph(SpecialGraphs:-MeringerGraph( ))
                    Graph 3: an undirected unweighted graph with 30 vertices and 75 edge(s)
EqualEntries( AdjacencyMatrix(
    Graph 2: an undirected unweighted graph with 30 vertices and 75 edge(s)),AdjacencyMatrix(
    Graph 3: an undirected unweighted graph with 30 vertices and 75 edge(s)))
                                        false
```

The new command **Eccentricity** computes the eccentricity of the graph at a specified vertex or, if not specified, computes the list of eccentricities at each vertex. The eccentricity of a vertex $v$ is the maximum graph distance between $v$ and any other vertex in the graph.

```
Eccentricity( SpecialGraphs:-HoffmanGraph( ) )
                    [3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4]
```

The maximum eccentricity over the entire graph is known as the *graph diameter* and can be computed with GraphTheory[Diameter] (also available in previous Maple versions).

```
Diameter( SpecialGraphs:-HoffmanGraph( ))
                                        4
```

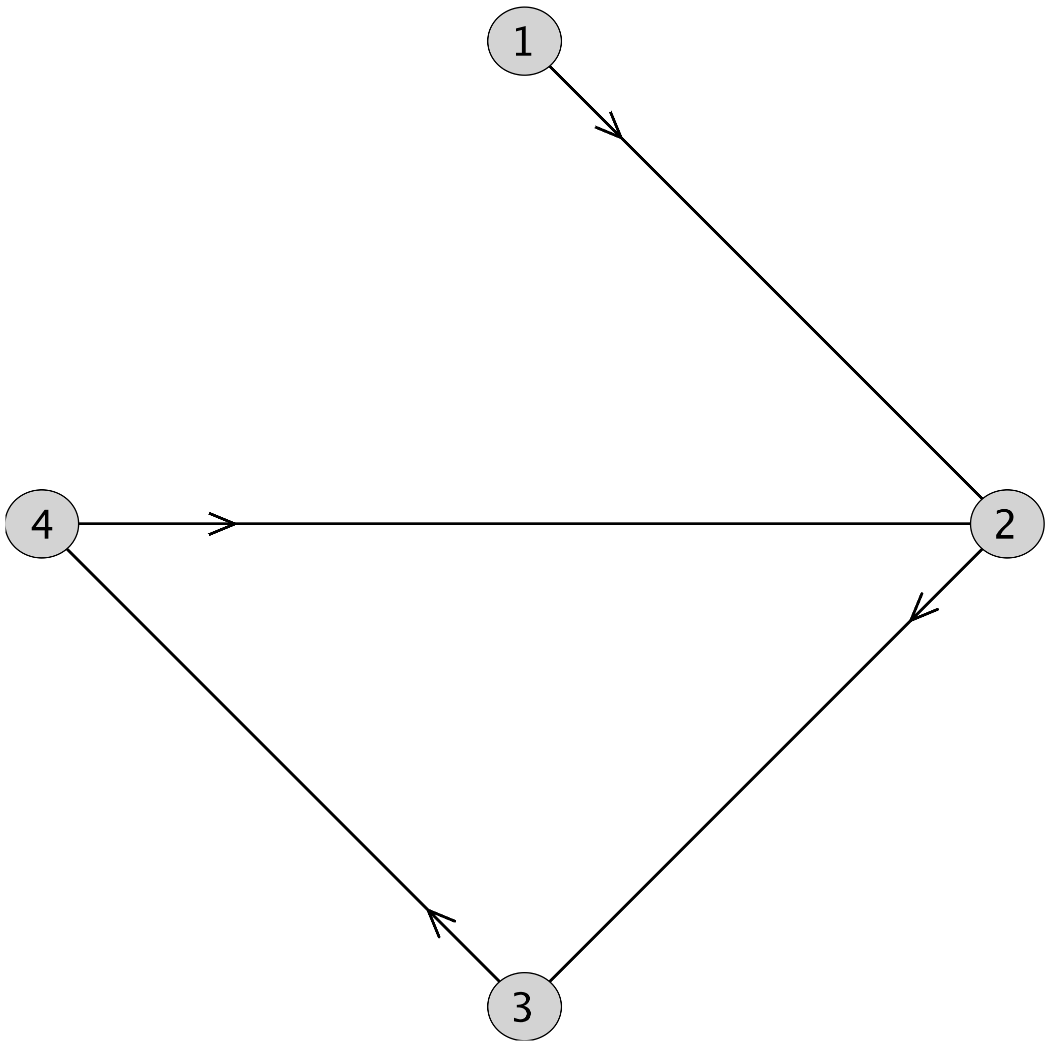
The minimum eccentricity over the entire graph is known as the *graph radius*, and it can be computed directly using the new command **Radius**:

```
Radius( SpecialGraphs:-HoffmanGraph( ))
                                        3
```

## Support for Digraph6 Format

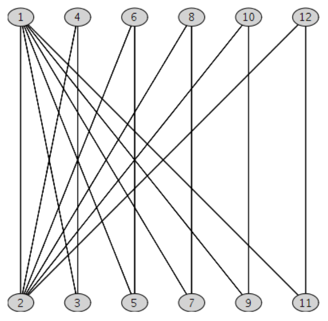
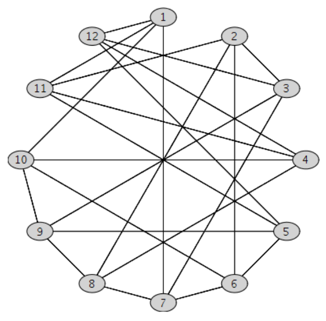
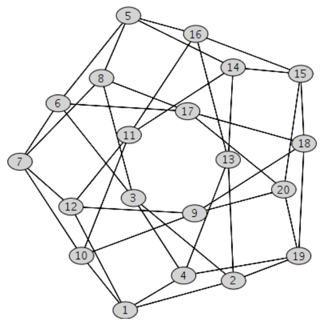
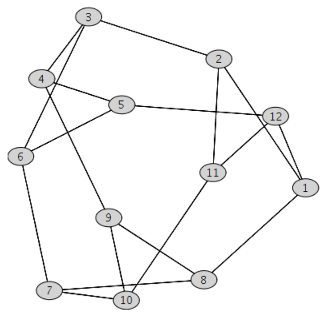
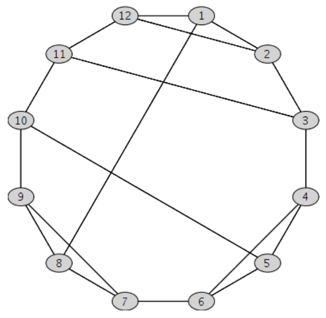
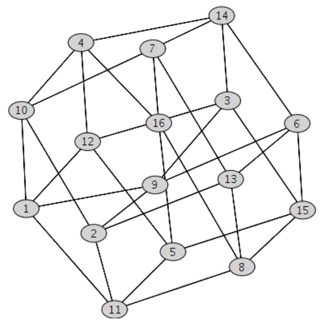
The general-purpose commands Import and Export as well as the GraphTheory commands GraphTheory[ImportGraph], GraphTheory[ExportGraph], and GraphTheory [ConvertGraph] now support the Digraph6 graph format. The Digraph6 format is a concise text-based format for serializing a directed graph.

```
> GraphTheory:-DrawGraph(Import("example/dgex.d6", base = datadir))
```



## New Special Graphs

The SpecialGraphs subpackage now includes built-in commands to generate the following special graphs:

| Book Graph | Chvatal Graph | Folkman Graph |
|---|---|---|
|  |  |  |
| **Franklin Graph** | **Frucht Graph** | **Hoffman Graph** |
|  |  |  |